Design and Analysis of a Hierarchical, Fault-Tolerant, Multicomputer Network

by

Feroze Badruddin Daud

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

June, 1996

DESIGN AND ANALYSIS OF A HIERARCHICAL, FAULT-TOLERANT, MULTICOMPUTER NETWORK

BY

FEROZE BADRUDDIN DAUD

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE
In
COMPUTER SCIENCE

JUNE 1996

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS
DHAHRAN, SAUDI ARABIA

COLLEGE OF GRADUATE STUDIES

This thesis, written by **FEROZE B. DAUD** under the direction of his Thesis Advisor and approved by his Thesis Committee, has been presented to and accepted by the Dean of the College of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** in **COMPUTER SCIENCE.**

THESIS COMMITTEE

Dr. Khalid Al-Tawil (Chairman)

Dr. Mostafa Abd-El-Barr (Co-Chairman)

Dr. Muhammad Al-Mulhem (Member)

Dr. Jarallah Al-Ghamdi (Member)

Department Chairman
Dr. Muhammad Al-Mulhem

Dean, College of Graduate Studies
Dr. Ala Al-Rabeh

Date

Dedicated

to

my Mother

Acknowledgements

Whatever is in the heavens and on earth,- let it declare the Praises and Glory of God: for He is the Exalted in Might, the Wise.

To Him belongs the dominion of the heavens and the earth: It is He Who gives Life and Death; and He has Power over all things.

He is the First and the Last, the Evident and the Immanent: and He has full knowledge of all things.

The Holy Quran, 57:1-3

Praise be to *Allah Subhanahu Wa Ta'la*, Lord of the Worlds, for giving me the strength, patience and understanding to undertake this work. I am indebted forever to God Almighty, for all the favors bestowed upon me.

I would like to acknowledge King Fahd University of Petroleum and Minerals for all the support extended during this research.

Thanks are due to my thesis committee chairman, Dr. Khalid Al-Tawil, and co-chairman, Dr. Mostafa Abd-El-Barr, for their encouragement and support throughout this research. I have really enjoyed working with them. I would also like to thank my thesis committee members, Dr. M. Al-Mulhem and Dr. Jarallah Al-Ghamdi for their helpful comments and suggestions.

Thanks are due to my friends and colleagues Sajjad, Nisar, Naseer, Kaleem, Nazeer, Ather, Raheem, Shoukat, Hashmi, Aijaz, Asad, Shakeel, Ashish and others for their support, and for making my stay in KFUPM an enjoyable one.

I would also like to thank my family: my Mother, Father, my Maternal Grandfather and Grandmother, Uncles and Aunts for all their patience, support and

strength, without which it would have been difficult to finish this work. All my achievements have been the results of their good wishes and encouragement. I especially thank my wife for her support, reassurance, understanding and patience all through this endeavor.

Contents

Acknowledgements iv

List of Figures x

List of Tables xi

Abstract(English) xii

Abstract(Arabic) xiii

1 Introduction 1
- 1.1 Topological Properties of the Hypercube . . . 2
- 1.2 Definitions and Background . . . 3
  - 1.2.1 Mean internodal distance . . . 3
  - 1.2.2 *LP* Product . . . 5
  - 1.2.3 *LP* Ratio . . . 6
  - 1.2.4 Cost . . . 6
- 1.3 Fault-Tolerance in Hypercubes . . . 7
- 1.4 Previous Work . . . 9
- 1.5 Motivation for the Thesis . . . 12
- 1.6 Objectives of this Work . . . 14
- 1.7 Organization of the Thesis . . . 15

2 Existing Fault-Tolerant Hypercube-based Networks 16
- 2.1 Single Fault-Tolerant Modular Scheme . . . 17
- 2.2 Modular Scheme based on Decoupling Networks . . . 19
- 2.3 Global Sparing Scheme based on Multiplexers . . . 22
- 2.4 The Fault-Tolerant Module . . . 24
- 2.5 The Perfect Embedding . . . 25
- 2.6 The 3-FTBB . . . 28
- 2.7 Concluding Remarks . . . 29

3 Existing Hierarchical Hypercube-based Networks 31
3.1 Binary Hypercube based Two-Level Network 32
3.2 The Hierarchical Hypercube 35
3.3 The Tightly Connected Interconnection Network 36
3.4 The Hierarchical Cubic Network 37
3.5 The Extended Hypercube 39
3.6 The dBCube 41
3.7 General Hierarchical Interconnection Networks 43
3.8 Hierarchical Network based on FTBBs 44
3.9 Concluding Remarks 45

4 Design and Analysis of a Hierarchical Interconnection Network 47
4.1 Design of the Basic Block 49
4.2 Design of the Level-Two Network 50
4.3 *HyperTorus* 51
4.4 Folded Hypercube based Architecture 53
4.5 Möbius Cube based Architecture 54
4.6 Results and Discussion 56
4.7 Properties of the Proposed Networks 60
4.8 Concluding Remarks 63

5 Design and Analysis of a Hierarchical, Fault-Tolerant Network 64
5.1 The Torus/4-cube(Perfect Embedding) 67
5.2 The Torus/Augmented 4-cube Scheme 69
5.3 The Torus/4cube-4spare Scheme 70
5.4 Results and Discussion 72
5.5 Properties of the Torus/4cube-4spare 74
5.6 Reconfiguration Strategy 79
5.7 Fault-Coverage Evaluation 83
5.7.1 Simulation Methodology 83
5.8 Cost and Performance Comparisons 84
5.9 Concluding Remarks 87

6 Reliability Evaluation of the Proposed Architecture 88
6.1 Reliability of the 4cube-4spare Basic Block 89
6.2 Reliability of Hierarchical Networks 93
6.2.1 Subtask reliability for a Torus 94
6.2.2 Subtask Reliability Comparisons 96
6.2.3 Results and Discussion 97
6.3 Network Reliability 98
6.3.1 NR bounds for the Hypercube 99

6.3.2 NR Bounds for the 4cube-4spare 100
6.4 Concluding Remarks 103

7 Routing Algorithms for the Proposed Architectures 104
7.1 The e-cube Algorithm 105
7.2 Fault-Tolerant Routing Algorithm for Binary Hypercubes 105
7.3 The 4cube-4spare Routing Algorithm 108
7.4 Routing in the *HyperTorus* 116
7.5 Concluding Remarks 120

8 Conclusions and Future Work 121
8.1 Conclusions 122
8.2 Future Work 123

Bibliography 125

Vita 130

List of Figures

1.1 A 3-dimensional hypercube. 2
1.2 An example graph to illustrate mean internodal distance calculation. 4

2.1 A subcube in Rennels' scheme. 18
2.2 Redundant multiprocessor for use in large array. 18
2.3 Decoupling networks used to interconnect fault-tolerant modules. . . 20
2.4 Using 2 decoupling networks to form a 2-dimensional fault-tolerant hypercube. 20
2.5 A fault-tolerant module with 4 primary nodes and k spare nodes. . . 21
2.6 Reconfiguring an FTM with 4 primary nodes and 3 spare nodes when a fault has occurred. 22
2.7 Switching logic used for reconfiguration. 24
2.8 A Fault-Tolerant Module of Yang's scheme. 25
2.9 An FTM based system with n=4, m=2, and p=1. 26
2.10 The architecture of P and S nodes. 27
2.11 Reconfiguration in Banerjee's scheme. 28
2.12 Hai and El-Barr's 3-FTBB. 29

3.1 A BH/BH HIN with a cluster size of 8. 34
3.2 A BH/BH-RS HIN with a cluster size of 8. 34
3.3 A 5-HHC. 36
3.4 A 2-Dimensional hypercube based 1TCN. 37
3.5 A Hierarchical Cubic Network with m=2 and n=2. 38
3.6 An EH(3,1) and a EH(3,2) constructed from 8 EH(3,1)s. 40
3.7 A dBCube(8,2) where the size of the dBGraph is 8 and each module is a hypercube of dimension 2. 42
3.8 A hierarchical interconnection network based on hypercubes. 44
3.9 A two-level hierarchical network using Hai's FTBB as the basic block. 45

4.1 A hierarchical interconnection network based on the torus, using hypercube based clusters. 51
4.2 Hypercube based proposed basic building block. 52
4.3 A Folded hypercube of order three. 53

4.4 A Möbius cube of dimension three. 55
4.5 Costs of the *HyperTorus*, Torus/Folded-cube and the Torus/Möbius-cube networks. 57
4.6 *LP* products of the *HyperTorus*, the Torus/Folded-cube, and the Torus/Möbius-cubenetworks. 59

5.1 The 4-cube(Perfect Embedding), a hypercube based fault-tolerant basic block. 68
5.2 An Augmented 4-cube constructed using 2-FTBBs. 71
5.3 The 4cube-4spare, a 4-cube based FTBB having four spares. 71
5.4 *LP* products of hierarchical fault-tolerant networks. 73
5.5 Costs of hierarchical fault-tolerant networks. 74
5.6 Interconnecting the FTBBs using a torus at level-two. 76
5.7 A 64 Node FTBB based hierarchical interconnection network. 77
5.8 A fault and the resultant spanning tree used for reconfiguration. . . . 80
5.9 Reconfiguration under link failures. 82
5.10 Fault-coverage comparison of the proposed 4cube-4spare with the hypercube and the torus. 84

6.1 The 4cube-4spare Fault-tolerant basic building block (FTBB). 89
6.2 Task Based Reliability for the FTBB and the 16-Cube. 91
6.3 Reliabilities of different fault-tolerant hierarchical networks. 97
6.4 Network reliability bounds for the hypercube and the 4cube-4spare . 103

7.1 The oblivious fault-tolerant hypercube routing algorithm. 107
7.2 Routing in a hypercube in the presence of faults. 108
7.3 Routing in case of network partitioning. 109
7.4 The oblivious fault-tolerant FTBB routing algorithm. 111
7.5 Routing in the FTBB in the presence of faults. 112
7.6 The routing algorithm executed by a spare node in a FTBB. 114
7.7 Routing when a spare node has replaced a primary node. 115
7.8 The *HyperTorus* routing algorithm. 118
7.9 Routing in the *HyperTorus*. 119

List of Tables

4.1 Performance measures of the *HyperTorus* network. 53
4.2 Performance measures of the Torus/Folded-cube network. 54
4.3 Performance measures of a hierarchical network with a 3-dimensional Möbius cube as the basic block. 56
4.4 Diameters of the *HyperTorus*, the Torus/Folded-cube and the Torus/Möbius-cube networks. 57
4.5 Mean internodal distances (M.I.D) and *LP* products of the *HyperTorus*, the Torus/Folded-cube and the Torus/Möbius-cube networks. 58
4.6 Topological comparison of hierarchical networks based on the Torus and Hypercube. 61
4.7 Topological comparison of recursive networks. 61
4.8 Topological comparison of hierarchical networks. 62

5.1 Performance measures of the Torus/4-cube(Perfect Embedding) network. 68
5.2 Performance measures of the Torus/Augmented 4-cube network. . . . 69
5.3 Performance measures of the Torus/4cube-4spare network. 72
5.4 Topological comparison of hierarchical fault-tolerant networks composed of the 4-cube(Perfect Embedding), the Augmented 4-cube, and the 4cube-4spare basic blocks. 79
5.5 Cost and performance of the Torus. 86
5.6 Cost and performance of the Hypercube. 86
5.7 Cost and performance of the Torus/4cube-4spare architecture. 86

THESIS ABSTRACT

Name: FEROZE B. DAUD

Title: DESIGN AND ANALYSIS OF
A HIERARCHICAL FAULT-TOLERANT
MULTICOMPUTER NETWORK

Degree: MASTER OF SCIENCE

Major Field: INFORMATION & COMPUTER SCIENCE

Date of Degree: MAY 1996

Many interconnection networks have been proposed in the literature, but none has wide ranging applicability. Hypercube networks are among the most commonly used interconnection network topologies. However, they have the disadvantages that link cost is exponential in network dimension and non-constant node degree. Torus networks have a fixed degree and low link cost, but a diameter which is linear in network size. Hierarchical interconnection networks provide a means to design networks which have low link cost, and which take advantage of the locality of communication existing in parallel applications. As network size increases, so does the probability of network failure. To safeguard against failure, spare nodes and/or links are incorporated into the network. In this work, we design a hierarchical interconnection network which has the desirable properties of the hypercube and the torus networks. The performance of this network is evaluated, and a routing algorithm is proposed. Fault-tolerance is incorporated into this network, and it's performance is compared to that of some other networks. A reconfiguration strategy is proposed, and reliability evaluation is undertaken. A Fault-Tolerant routing algorithm is proposed for this architecture.

Keywords: Hypercube, Torus, Hierarchical Interconnection Networks, Fault-Tolerance, Routing, Reliability, Reconfiguration, HyperTorus.

King Fahd University of Petroleum and Minerals, Dhahran.

June 1996

خلاصة الرسالة

اسم الطالب الكامل : فيروز داوود

عنوان الدراسة : تصميم وتحليل الشبكات الهيكلية متعددة الحاسبات ذات خاصية تحمل الأخطاء

الدرجة : ماجستير في العلوم

التخصص : معلومات وعلوم الحاسب الآلي

تاريخ الشهادة : يونيو ١٩٩٦

لقد تم اقتراح العديد من الشبكات المتصلة فيما بينها في الأبحاث المنشورة، ولكن ليس من بينها ما يتصف بقابليته للتطبيق علي نطاق واسع. الشبكات المكعبية هي أحد التوزيعات الأكثر انتشارا. عيوب هذه الشبكات أن تكلفة الوصلة تتزايد بشكل أسّي بالنسبة لأبعاد الشبكة وأن ودرجة العقدة غير ثابتة. شبكات تورس تتمتع بعقد ذات درجة ثابتة وانخفاض تكلفة الوصلة، ولكن قطرها ذا تناسب خطي مع حجم الشبكة. الشبكات الهيكلية المتصلة فيما بينها تعطي إمكانية تصميم شبكات ذات تكلفة منخفضة للوصلة و تستغل خاصية محلية الاتصال الموجودة في التطبيقات المتوازية. كلما ازداد حجم الشبكة، كلما زاد احتمال إخفاق الشبكة ولكي تتم حماية الشبكات من الإخفاق، يمكن إضافة عقد احتياطية و/أو وصلات إلى الشبكة. في هذه الرسالة، نقوم بتصميم شبكة هيكلية متصلة فيما بينها وتتمتع بالخواص المرغوبة في الشبكات المكعبة و شبكات تورس. لقد تم تقييم أداء هذه الشبكة وتم اقتراح خوارزم توجيه. لقد تم إدخال خاصية تحمل الأخطاء في هذه الشبكة وتم مقارنة أدائها بأداء بعض الشبكات الأخرى. تم اقتراح استراتيجية لإعادة التوزيع مع تقييم إمكانية الاعتماد علي الشبكة ومن ثم، تم اقتراح خوارزم توجيه لتحمل الأخطاء في هذا التصميم.

كلمات البحث :المكعب الزائد، تورس، الشبكات الهيكلية المتصلة فيما بينها، تحمل الأخطاء، توجيه، إمكانية الاعتماد، إعادة توزيع، تورس الزائد.

جامعة الملك فهد للبترول والمعادن

الظهران ، المملكة العربية السعودية

يونيو ١٩٩٦

Chapter 1

Introduction

Distributed computing has offered a cost effective alternative for performing massively parallel computations by interconnecting a number of processors having local memory. These processors communicate through message passing. Hypercubes [1] belong to a class of distributed memory MIMD computers. A Hypercube offers a small diameter and less number of links per node. It is highly regular, symmetric and recursive in nature. Among the desirable features of hypercubes is their ability to efficiently simulate any bounded degree network. A number of commercial multicomputers have been built using the hypercube topology. The binary hypercube interconnection scheme has been used in the Connection Machine, Cosmic Cube [2], and commercial systems such as Intel iPSC and the NCUBE/ten [3].

Multicomputers used in mission critical applications require a very high reliability and availability. As the number of nodes in a given architecture increases, so does the probability of failure. One of the disadvantages of a hypercube is that a few node failures can destroy the whole network. Thus, it becomes important to incorporate fault tolerance into the system.

Many approaches to provide fault-tolerance in hypercubes have been proposed in the literature. Some are hardware based and some are software based. One approach is to identify the maximal size subcube in a faulty hypercube and run the application on it. Another approach is to redistribute the tasks of faulty processors to non-faulty processors.

In this chapter, we define the hypercube topology in Section 1.1 and then give a brief introduction to it's properties. In Section 1.3 we give an introduction to fault-tolerance in hypercube networks. We conclude by giving an organization of the thesis in Section 1.7.

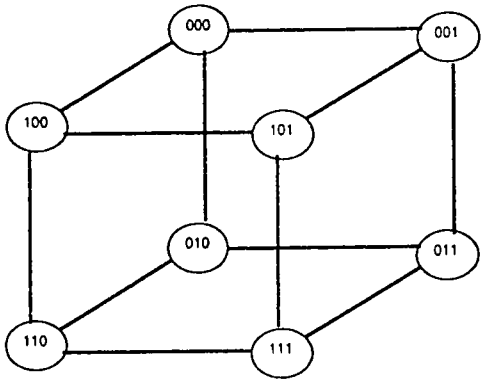

Figure 1.1: A 3-dimensional hypercube.

1.1 Topological Properties of the Hypercube

A d−dimensional hypercube is an undirected graph consisting of $N = 2^d$ nodes. The nodes are labeled {0,1,2,...,N-1}. Each node has a d bit address $(i_{d-1}, i_{d-2}, i_{d-3}, ..., i_1, i_0)$. A 3-cube is shown in Figure 1.1. There is a link between two nodes if and only if their binary addresses differ in one and only one bit. There are $d \cdot 2^{d-1}$ links in

a d-dimensional hypercube. A link is said to span dimension i if the endpoints of the link have binary addresses which differ in the ith bit position. The minimum distance (number of links) between two nodes is equal to the Hamming distance between their addresses. The maximum distance between two nodes in a non-faulty hypercube is equal to the maximum of the Hamming distances between any two node addresses and is equal to d. The mean internodal distance between any two nodes in the network is equal to $d/2$. Each node has direct links to d other nodes. A $j-$subcube of a $d-$cube is defined as a subgraph consisting of 2^j nodes obtained by choosing values (0's and 1's) for $d-j$ dimensions such that all node addresses in the $j-$subcube have same values for these $d-j$ dimensions [4, 5]. For example, 0xx and 1xx are two disjoint 2−cubes in a 3−cube, each containing four nodes where 'x' means don't care. In general we have ${}^dC_{d-j}$ sets of disjoint $j-$subcubes in a $d-$cube. As an example, in a *3*-cube, there are three sets of disjoint *2*-cubes - {{0xx,1xx}, {x0x,x1x}, {xx0,xx1}}

1.2 Definitions and Background

In this section, we define the performance measures used in the thesis. The performance measures used are cost, mean internodal distance [6, 7] and LP ratio [8].

1.2.1 Mean internodal distance

Mean internodal distance is a fundamental property of a topology. It is the expected number of link traversals needed by a typical message to reach the destination. It is a better indicator of average message delay than diameter [9]. Diameter is the

maximum inter-node distance in a network. Diameter only indicates the worst case message distance in a network. It is more informative to know the expected distance travelled by a message, which is a more realistic indicator of delay in a network.

Definition 1 *Let h_{ik} be the number of nodes reachable in k hops starting at node i. Let d_i be the number of hops necessary to reach the nodes farthest from node i. Then the mean internodal distance of node i from the rest of the network is defined as:*

$$\bar{h_i} = \frac{\sum_{k=1}^{d_i} k h_{ik}}{\sum_{k=1}^{d_i} h_{ik}} \tag{1.1}$$

The mean internodal distance P for the network is then

$$P = \frac{1}{N} \sum_{i=1}^{N} \bar{h_i} \tag{1.2}$$

where N is the number of nodes in the network.

Example: In Figure 1.2 we have a small network. The distance of node *1* from

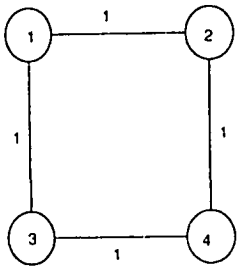

Figure 1.2: An example graph to illustrate mean internodal distance calculation.

the rest of the network can be calculated as follows:

$$h_{11} = 2$$

$$
\begin{aligned}
h_{12} &= 1 \\
d_1 &= 2 \\
\bar{h_1} &= \frac{1 \times h_{11} + 2 \times h_{12}}{h_{11} + h_{12}} \\
\bar{h_1} &= \frac{1 \times 2 + 2 \times 1}{1 + 2} \\
\bar{h_1} &= 4/3
\end{aligned}
$$

Similarly,

$$
\begin{aligned}
\bar{h_2} &= 4/3 \\
\bar{h_3} &= 4/3 \\
\bar{h_4} &= 4/3
\end{aligned}
$$

The mean internodal distance P for the network is then

$$
\begin{aligned}
P &= \frac{(4 \times 4/3)}{4} \\
P &= 4/3
\end{aligned}
$$

This indicates that on an average, 4/3 links will be traversed by any message in this network to reach it's destination.

1.2.2 LP Product

In general, trying to minimize the mean internodal distance P results in an increase in the number of links L and vice versa. Therefore, a useful measure is $L \times P$ which

should be minimized. The LP product is a cost-benefit ratio, with L representing the cost of the network, and $1/P$ representing the benefit [8].

Definition 2 LP *Product is defined as the product of the mean internodal distance P and the total number of links L.*

For the network in Figure 1.2, the LP product is $4 * 4/3 = 16/3$.

1.2.3 LP Ratio

When comparing the performance of a given interconnection network with that of a reference network, it is useful to divide the LP product of the given network by that of the reference network. If the result (i.e., LP ratio) is less than one, it means that there is an improvement in performance (with respect to the LP ratio) associated with the given network.

Definition 3 LP *Ratio is defined as the ratio of the* LP *product of the proposed network to the* LP *product of the reference network.*

$$LPratio = \frac{L_{\text{proposed}} \times P_{\text{proposed}}}{L_{\text{reference}} \times P_{\text{reference}}} \tag{1.3}$$

1.2.4 Cost

Diameter is defined as the maximum internode distance in an architecture. Usually, there is a tradeoff between diameter and degree of a communication network. The diameter can be decreased by adding more links, and hence increasing the degree of

the network. But this leads to an increase in link cost. Therefore, the product of degree and diameter is a useful parameter to obtain the cost of a network.

Definition 4 *The cost of a communication network is defined as the product of the maximum degree of a node and the diameter of the network.*

$$cost = degree \times diameter$$

1.3 Fault-Tolerance in Hypercubes

The hypercube has a regular structure, which can be destroyed by node or link failures and may lead to performance degradation or system failure. To avoid this, it becomes necessary to incorporate fault-tolerance into the system.

Fault-tolerance is the property of a system to continue working in the presence of faults. The effect of faults can be neutralized by incorporating redundancy into the system. Fault-tolerance can be achieved by many techniques. One approach is to detect the faulty component and reconfigure the network to avoid that faulty component. This means that the network should have some form of redundancy. This redundancy can be incorporated in several ways.

- *Hardware Redundancy* is the addition of extra hardware (nodes and/or links) to tolerate faults.

- *Software Redundancy* is the addition of software to detect and possibly recover from faults. For example, a robust routing algorithm might route information around faulty components.

- *Information Redundancy* is the addition of extra information beyond that required to achieve the given function. Error detecting and correcting codes are used as forms of information redundancy.

Fault-tolerant systems can be specified as highly available or highly reliable. The *Availability* $A(t)$ of a system as a function of time is defined as the probability that the system is operational at time t. In the limiting case ($t \rightarrow \infty$), it expresses the expected fraction of time the system is able to perform useful computations.

The *Reliability* $R(t)$ of a system as a function of time, is defined as the conditional probability that the system has survived the interval $[0, t]$ given that it was operational at time $t = 0$.

A conventional reliability modelling approach considers the system to be a black box, and assumes that the links and/or nodes can fail independently with known probabilities. The reliability of a hypercube based multicomputer system is generally computed by considering the following measures.

1. **Network Reliability (NR)**: The system works as long as all nodes in the system are working and connected.

2. **Task-Based Reliability (TBR)**: The system works as long as some minimum number of connected nodes are available in the system for task execution.

3. **Terminal Reliability (TR)**: The system works as long as two specified nodes are working and connected.

4. **Subcube Reliability (SR)**: The system works as long as some functional minimum degree subcube exists.

Fault-tolerance can be measured in terms of *network connectivity*. Network connectivity measures the ability of a network to continue operation despite failed network components (i.e., nodes and links). Informally, network connectivity is the minimum number of network components that must fail to partition the network into two disjoint subnetworks. This leads us to *link connectivity* of a network, which is defined as the minimum number of links that must be removed from the network in order to partition the network into two subnetworks. The *node connectivity* is defined analogously.

Mean internodal distance is one of the metrics used to characterize network connectivity. It is defined as the average number of link traversals needed by a typical message in the network. The minimum necessary aggregate link capacity in an interconnection network is directly proportional to the mean distance between the nodes [6, 7]. Thus, mean internodal distance is an important network characteristic.

Several fault-tolerant hypercube architectures have been proposed in the literature with their reliability analysis. In [10], terminal reliability and network reliability of hypercube architectures have been analyzed, and tighter bounds have been obtained.

1.4 Previous Work

The probability of failure in interconnection networks is proportional to the size of the network. Hence it is important to incorporate fault-tolerance into such networks. Fault-tolerance can be incorporated by adding hardware or software redundancy. In [11], extra links were added between pairs of nodes in hypercubes to maximize the improvement of performance measures under various traffic distributions. Rennels

[12] used spare nodes to tolerate node failures. Nodes are built with extra ports to communicate with the spares. Crossbar switches are used for reconfiguration. Chau et. al. [13] proposed an architecture based on modular construction. Decoupling networks were used to realize inter-modular and intra-modular connections in the architecture. Each module consists of m active nodes and k spare nodes. Sultan and Melhem [14] introduced a scheme using hardware switches. This scheme does not have the overhead of switching processor states. Yang et al [15] proposed another fault-tolerant hypercube architecture using fault-tolerant modules (FTMs) as the basic building blocks. A spare node in a FTM can be used to replace a faulty node in any other FTM using spare-sharing links. This architecture tolerates node as well as link failures. It has been shown in [16, 17, 18] that typical applications running on a hypercube use the DMA/router less than 10% of the time. It was deduced from this observation that a router could be shared between two CPUs [16]. Nodes having a spare CPU and Memory were embedded into the hypercube such that they were adjacent to atleast one primary node. The spare CPU and DMA could then could replace a failure in any adjacent node. Hai [19] proposed a fault-tolerant hypercube. In this scheme, one spare node was embedded in a hypercube of dimension d. The spare was connected to all the nodes of the hypercube. Node as well as link failures are tolerated by this scheme.

As system size increases, the number of links needed with structures like hypercubes grows exponentially. Also, there is some locality of communication between nodes in an interconnection network. Hierarchical interconnection networks (HINs) try to take advantage of these factors. These networks have multiple levels, where lower level networks provide local communication and higher level networks facili-

tate remote communication. Many hierarchical networks based on hypercubes have been proposed in the literature. In [8, 20, 21], a two level fault-tolerant HIN is proposed in which the level one network is a collection of hypercube clusters, and the level two network is a 2-cube. Fault-tolerance is provided by either duplicating the level two network or by providing spare interface nodes. Qutaibah [22] proposed a three level Hierarchical Hypercube (HHC). The degree of the nodes is much less than the normal hypercube. Also the diameter of the architecture is logarithmic. The Extended Hypercube [23] is a hierarchical architecture consisting of a k-cube of processor elements connected to a network controller (NC). It is a truly expansive, recursive structure with a constant predefined building block. The utilization factor (ratio of the time spent doing computation to the time spent doing communication) of the Extended Hypercube is considerably larger than that of a hypercube. In [24], a compound network consisting of a deBruijn graph and a hypercube is proposed. It has the basic characteristics of the hypercube and a smaller effective diameter. The Hyper-deBruijn Network [25] is a combination of hypercube and deBruijn network. It possesses logarithmic diameter, optimal connectivity and simple routing algorithms amenable to networks with faults. In [26], a method for constructing hierarchical interconnection networks is presented. When applied to meshes and hypercubes, it results in networks with optimal connectivity, high bisection width (defined as the least number of links that have to be removed in order to partition the network into two halves, each having half the nodes as the original network), low degree, diameter and cost. Potlapalli et al [27] proposed a dynamic scheme for constructing HINs using multistage interconnection networks. Ghose and Desai [28] presented a new interconnection network based on the hypercube. It has about

three-fourths the diameter of a comparable hypercube, using about half as many links per node. Hai [19] introduced a hierarchical hypercube using his FTBB as a level-one network. The reliability of this network was shown to be better than the hypercube and Dandamudi's scheme [8].

1.5 Motivation for the Thesis

This section focuses on the motivations for the thesis. Fault-tolerance in hypercubes can be incorporated in two ways, through software or hardware. In the hardware approach, extra components (nodes and/or links) are added to the hypercube to replace the failed components. In this approach, spares can be added to the hypercube in two ways, using either modular sparing or global sparing. In *modular sparing*, a module is built which comprises of a hypercube and spares. Many such modules are connected together in order to realize a hypercube of the required dimension. In each module, the spares are connected to the primary nodes through extra links or other hardware such as switches, crossbars etc. In *modular sparing*, a spare in one module cannot replace failed nodes in another module. Thus, schemes based on this approach have a simple reconfiguration algorithm. However, there might be less than full spare utilization. In *global sparing*, the spares in one module can be used to replace the failed nodes in another module. This leads to better spare utilization. However, there is a high hardware overhead entailed, because the primary nodes have to be connected to all the spare nodes. The reconfiguration strategy is more elaborate in these schemes as compared to those used in the modular sparing schemes.

A majority of the fault-tolerant hypercubes proposed in the literature only

tolerate node failures [12, 13, 14]. These schemes involve a lot of hardware overhead in the form of multiplexers [14], crossbar switches [12, 15] or decoupling networks [13]. In Banerjee's scheme [16, 17], the architecture of the nodes is not homogenous. These schemes sought to preserve the hypercube topology in the presence of faults, at the cost of using extra hardware such as switches, spare ports in nodes, etc. The reliability of these extra components was not taken into account when system reliability analysis was performed. In addition, there have been some inherent difficulties with recursively-constructed hypercube based structures. First, the complexity of the node has to be changed if the size of the network is increased [13, 14, 15, 16, 19]. Second, the number of links became a limiting factor.

In recent years, hierarchical networks have attracted a lot of interest, because they provide a framework to design networks with reduced link cost. Also, they take advantage of the inherent synergy among communicating tasks in a hypercube. In other words, they take advantage of the locality of communication that exists in parallel applications in general. However, fault-tolerance has not been considered in some of these architectures [22, 23, 26, 27, 28]. Also, the node complexity increases with network size and they are not easily expandable [19, 22, 26, 27, 28, 29]. Networks based on the deBruijn graph [24, 25] are irregular. Routing algorithms for these architectures are very complex. Also, it is very difficult to map algorithms to these architectures. Tree based schemes [23, 27] have the disadvantage of high message density towards the root and low fault-tolerance.

Networks like mesh, torus etc. have a fixed degree which is independent of the size of the network. The torus has the additional advantage of using symmetric nodes. However, these architectures do not have logarithmic diameter. Hypercube

networks have logarithmic diameter, but node degree increases with network size. Thus, it would be desirable to have the simplicity of bounded degree networks and the versatility of the hypercube networks.

This motivates us to propose a new fault-tolerant, hierarchical interconnection network based on the hypercube and the torus. It consists of hypercube based fault-tolerant basic building blocks (FTBBs), interconnected using a torus. This architecture has modular sparing. The FTBB of our scheme is a four dimensional binary hypercube to which spare nodes and links have been attached for fault-tolerance.

1.6 Objectives of this Work

This work focuses on the problem of incorporating fault-tolerance in hypercube based hierarchical interconnection networks. One of our main objectives is to study the advantages and the disadvantages of existing fault-tolerant architectures, hierarchical and non-hierarchical. Then, we will experiment with different clusters for possible incorporation into a hierarchical network. A performance comparison among these hierarchical architectures will be made, with the objective of coming up with one which is expandable. The architectures considered in this study will be hierarchical in nature, and consist of two levels. The level-one network consists of hypercube based clusters, and the level-two network consists of a torus network which interconnects the clusters. At the end of this study, we will come up with a candidate architecture. The same procedure will be followed to design a fault-tolerant hierarchical interconnection network. This will result in the selection of a candidate architecture. It's resilience to faults and it's performance compared to

that of the hypercube and the torus will be studied. The reliability of this architecture will be analyzed. A fault-tolerant routing algorithm will be developed for this architecture. We shall investigate the reliability of this architecture using different analytical models. The results obtained will be compared to those of other architectures.

1.7 Organization of the Thesis

The thesis is organized as follows. Chapter 2 discusses some previous proposals for fault-tolerant hypercubes. Some hierarchical networks proposed in the literature are discussed in Chapter 3. Different hierarchical networks are evaluated from the point of view of performance in Chapter 4. The aim will be to come up with a candidate architecture that has the desirable properties of expandability and fault-tolerance. This will be followed by an investigation of it's properties in Chapter 5. Reliability evaluation of the architecture is presented in Chapter 6. This chapter presents the reliability of the basic block, as well as that of hierarchical networks composed of the proposed basic block. Chapter 7 presents a routing algorithm for the proposed architecture. We conclude by summarizing our contributions, and give some directions for future research in Chapter 8.

Chapter 2

Existing Fault-Tolerant Hypercube-based Networks

The probability of failure in interconnection networks is proportional to the size of the network. Hence it is important to incorporate fault-tolerance into such networks. Fault-tolerance can be incorporated by adding hardware or software redundancy. The former method, in which extra nodes and or links are added to increase fault-tolerance, has been studied by many researchers. In this chapter, we shall give an overview of some of these schemes.

This chapter is organized as follows. We start by reviewing existing fault-tolerant hypercube architectures. We shall study the proposals of Rennels [12], Chau et. al [13], Sultan and Melhem [14], Yang et al [15], Banerjee and Peercy [16] and Hai [19]. Section 2.7 concludes this chapter.

2.1 Single Fault-Tolerant Modular Scheme

Rennels [12] suggested two schemes that use spare nodes to tolerate node failures in a hypercube. In both schemes, the nodes are built with extra ports to communicate with spares. In the first scheme, an $n-$cube comprises of 2^s subcubes, each containing 2^m nodes, where $s + m = n$. One spare node is provided for each m-cube, and the nodes are connected through their extra port to the spare. Whenever a primary node fails, the spare is connected to the m neighboring nodes in the subcube and to the s neighbors in other neighboring subcubes. Two crossbar switches per spare are used for reconfiguration. The Connection Crossbar (CCB) has $2^m + s$ inputs and n outputs. The outputs are connected to the spare. Out of the $2^m + s$ inputs, 2^m are connected to the primary processors. Each of the remaining s inputs comes from one output of the Relay Crossbar (RCB) of each of the other s subcubes to which the module is connected. The RCB has 2^m inputs and s outputs. The inputs come from all the nodes in the subcube and each output goes to one of the s subcubes to which the subcube is connected. On failure in a subcube, it's CCB connects the spare processor to the m neighbors of the failed processor. The RCB in each neighboring subcube reconfigures so that the neighbor of a failed processor is now connected to the replacement of the failed processor. The various components of this scheme are shown in Figure 2.1.

The reliability of this scheme is calculated as follows. Each subcube of order m can tolerant at most one fault. There are 2^s such subcubes. Therefore the system reliability of this scheme is:

$$RB_{m,s} = [r^{2^m} + 2^m r^{2^m}(1 - r)c]^{2^s} \tag{2.1}$$

where r is the reliability of each node, and c is the fault coverage. Fault coverage is defined as the probability of successful reconfiguration in the presence of faults.

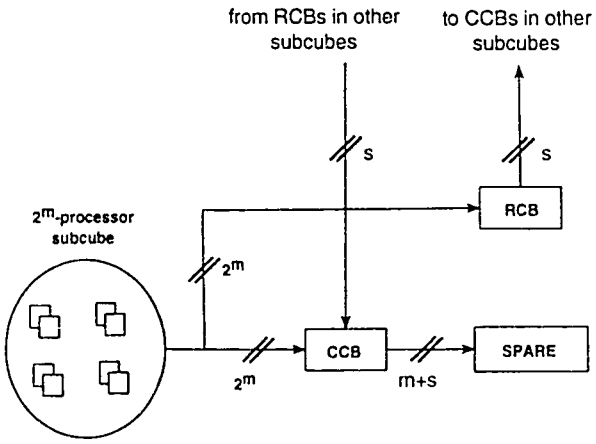

Figure 2.1: A subcube in Rennels' scheme.

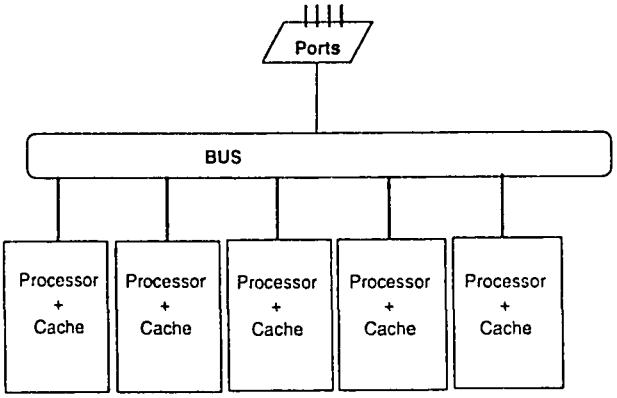

Figure 2.2: Redundant multiprocessor for use in large array.

For high reliability systems, another scheme is proposed in which each module is made internally redundant. The redundancy is in the form of spare memory modules, processors etc. One such module is shown in Figure 2.2. Four processors

and a spare are connected by a high speed bus and packaged as a multiprocessor. Four ports are provided to communicate with other multiprocessors. To realize a 64-node cube, for example, 16 multiprocessors are interconnected as a 4-cube. This scheme can be applied recursively to obtain higher order cubes. This scheme is highly reliable and provides better spare sharing.

2.2 Modular Scheme based on Decoupling Networks

Chau et al [13] proposed a fault-tolerant-hypercube architecture based on modular construction. Each fault-tolerant module (FTM) consists of 2^m active nodes and k spare nodes. Decoupling networks are used to realize the connections within and among the FTMs. Figure 2.3 shows k levels of decoupling networks being used to connect 2^m primary nodes (m=2 in the figure) and k spare nodes from one FTM to another. To realize intramodular connections, m groups of k level decoupling networks are used. Figure 2.4 shows how a 2-dimensional hypercube can be constructed using two decoupling networks.

When $m = 2$ and with k spares, the intramodular connections can be realized using soft switches as shown in Figure 2.5. When a fault occurs, the soft switches bypass the failed node. At the same time, the decoupling networks are reconfigured, as shown in Figure 2.6 so that the hypercube structure is maintained. This scheme is generalized for global sparing, where $m = n$ and the entire network is a fault-tolerant module.

In the local sparing scheme, where the n-cube consists of FTMs having 2^m

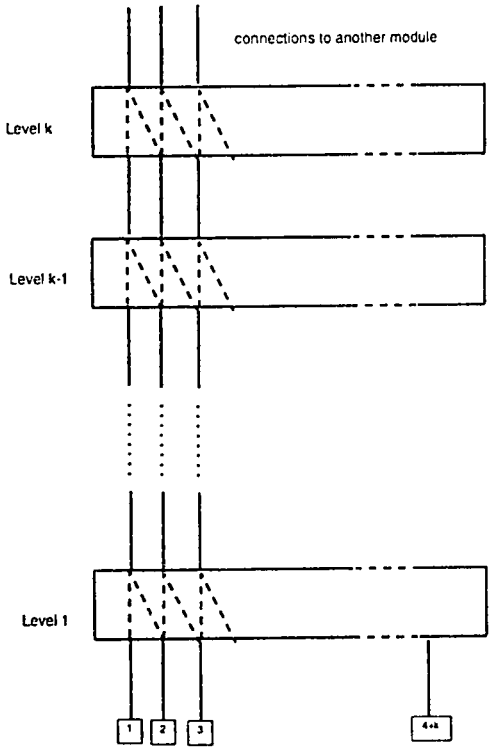

Figure 2.3: Decoupling networks used to interconnect fault-tolerant modules.

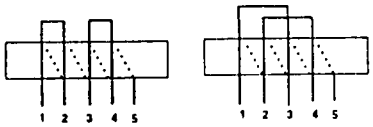

Figure 2.4: Using 2 decoupling networks to form a 2-dimensional fault-tolerant hypercube.

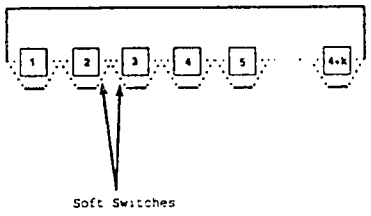

Figure 2.5: A fault-tolerant module with 4 primary nodes and k spare nodes.

primary nodes and k spare nodes, the reliability of a module is given by:

$$RM_{m,k} = RM_{m,k-1} + \begin{pmatrix} 2^m + k - 1 \\ k \end{pmatrix} r^{2^m}(1-r)^k c^k \tag{2.2}$$

where r is the reliability of a node. The reliability of an n-dimensional hypercube with 2^m active processors and k spares in each module is given by:

$$RS_{n,m,k} = (RM_{m,k})^{2^{n-m}} \tag{2.3}$$

The global sparing scheme utilizes lesser number of spares as compared to Rennels basic scheme to achieve the same level of reliability. Chau [13] showed that when $n \leq 8$, global sparing is preferable to modular sparing.

This scheme takes longer to reconfigure in the presence of faults because on average, the states of half the nodes will have to be shifted to neighboring nodes. Furthermore, this scheme does not account for link failures. Only the system reliability under node failures has been analyzed.

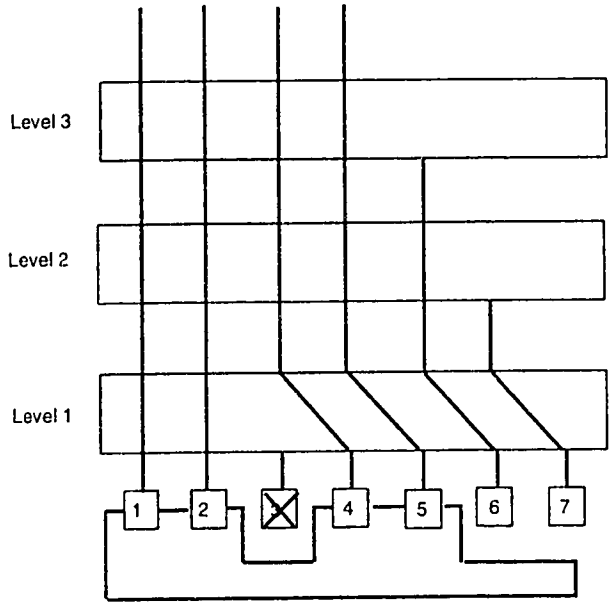

Figure 2.6: Reconfiguring an FTM with 4 primary nodes and 3 spare nodes when a fault has occurred.

2.3 Global Sparing Scheme based on Multiplexers

Sultan and Melhem [14] introduced two methodologies for reconfiguration in fault-tolerant modules. The first scheme uses hardware switches to reconfigure in the presence of faults, and provides full spare utilization. As an example, consider an FTBB with M primary nodes $P_1, P_2, ...P_M$ and K spare nodes $S_1, S_2, ..., S_K$. Multiplexers and demultiplexers are used to implement the switching logic for reconfiguration. A 1-to-(K+1) demultiplexer is used for each P_j to divert, when needed, the links of P_j to any S_i. Also, an M-to-1 multiplexer is used for each S_i to connect it to the M neighbors of the failed node. The switching logic is shown in Figure 2.7. If a primary node P_j is non-faulty, then the demultiplexer is set to select the 0th

output. In case P_j is faulty, the replacement of P_j by S_i requires that the demultiplexer associated with P_j be set to it's ith output and the multiplexer associated with S_i be set to select it's jth input. The scheme can also support spare failures and does not have the overhead of switching processor states, which is present in Chau's scheme. This leads to significant reduction in reconfiguration time and overhead. Also, Chau's scheme requires $2^{n-m}nm$ more switches as compared to this scheme to achieve the same level of reliability while using the same number of spares per module. Assuming the reliability of a node to be r, the reliability of this scheme is calculated as follows:

$$R_{FTBB} = \sum_{i=0}^{k} \binom{m+k}{i} r^{m+k-i}(1-r)^i \tag{2.4}$$

$$R_{sys} = R_{FTBB}^{2^{n-m}} \tag{2.5}$$

In the second technique, reconfiguration is done using a two-phase routing algorithm. In the first phase, the message is routed to the destination FTBB, i.e. the FTBB which contains the destination node. In the second phase, the message is routed to the destination node within that FTBB. It is assumed that the failure of a node is the failure of the processor, the router and the links of the node. It is also assumed that the spare which replaces a failed node inherits it's address. The routing algorithm is distributed and requires only the neighbors of the faulty nodes to know about the faults.

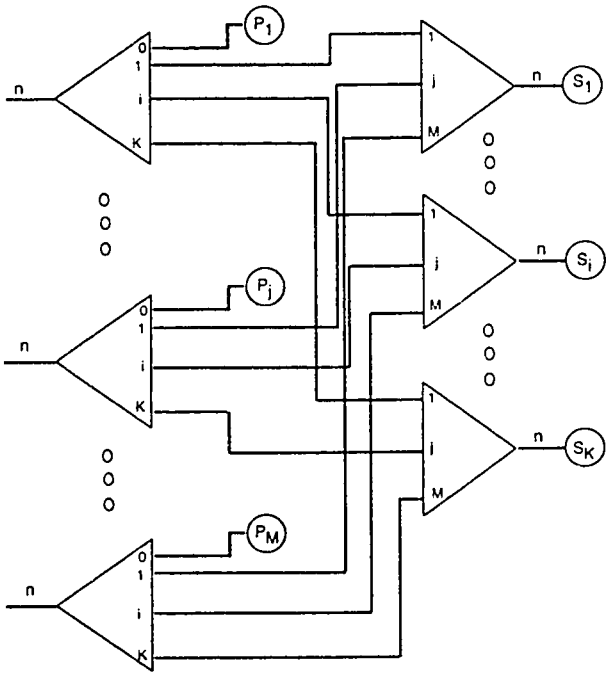

Figure 2.7: Switching logic used for reconfiguration.

2.4 The Fault-Tolerant Module

Another scheme using fault-tolerant modules (FTMs) was proposed by Yang et al [15]. This scheme provides global sparing. Each module contains 2^m active nodes and p local spare nodes. An Internal Switching Connector (ISC) connects 2^m out of a total of $2^m + p$ nodes in each FTM to form an m-cube topology. The FTM of this scheme is shown in Figure 2.8. The FTMs are interconnected using $(n - m)$ External Switching Connectors (ESCs) per module. To provide global sparing, the 2^{n-m} FTMs are also connected as a ring to allow the idle local spares in one module to be used by another. This is done using Spare Sharing Links (SSLs), as shown in Figure 2.9. An FTM with k $(p < k \leq 2p)$ faulty nodes can use it's own local spare nodes as well as the remote spares supplied from other FTM's to replace the

faulty nodes. Thus, the fault-tolerance capacity of each FTM is improved, and the overall system reliability is enhanced. Node as well as link failures are tolerated in this scheme.

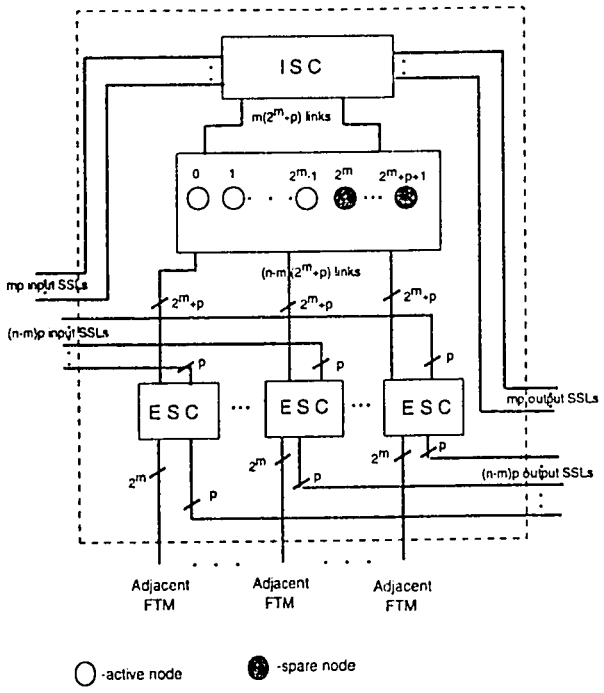

Figure 2.8: A Fault-Tolerant Module of Yang's scheme.

2.5 The Perfect Embedding

Banerjee and Peercy proposed two fault-tolerant hypercube schemes [16]. The first scheme called the *node spare scheme* uses spare nodes attached to specific nodes in the cube using novel embedding techniques. In this scheme there are two kinds of nodes, the P nodes and S nodes. The P and the S nodes are embedded in the hypercube in such a way that each P node is adjacent to exactly one S node in the

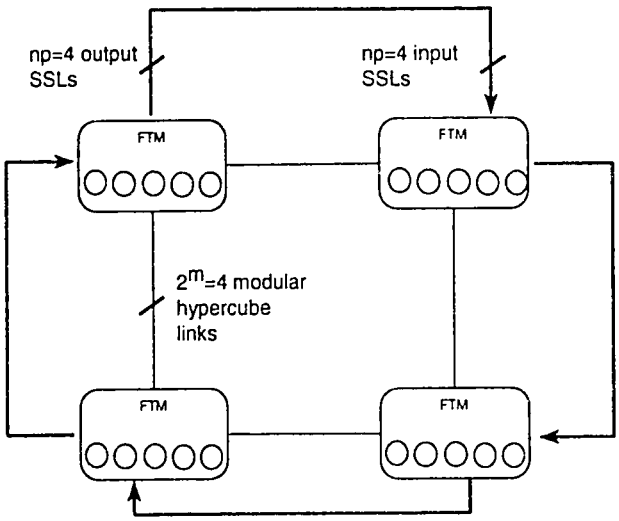

Figure 2.9: An FTM based system with n=4, m=2, and p=1.

cube, as shown in Figure 2.10a. This is called *perfect embedding.* The architectures of P nodes and S nodes are shown in Figure 2.10b and Figure 2.10c respectively. The P node consists of a computation processor (CPU) connected through an internal bus to a local memory and message routing logic consisting of the DMA unit and a $(d+1)\times(d+1)$ crossbar switch for a 2^d processor hypercube. The S-node consists of two copies of CPU and memory connected to two internal busses. The two processing units share the DMA and message routing logic. One of the processors is active under normal conditions; the other is a standby spare. When any processing element fails, either within the S node or in a nearby P node, the spare processor/memory/bus from the corresponding S node replaces it. Figure 2.11 shows how reconfiguration is done in a 4-cube in the presence of faults. An algorithm for reconfiguration in the presence of primary node failures is given in [16]. The reliability of this scheme is good for relatively small number of faults. As the number of faults increases, the reliability goes down, as each primary node is covered by only one spare (see

Figure 2.11).

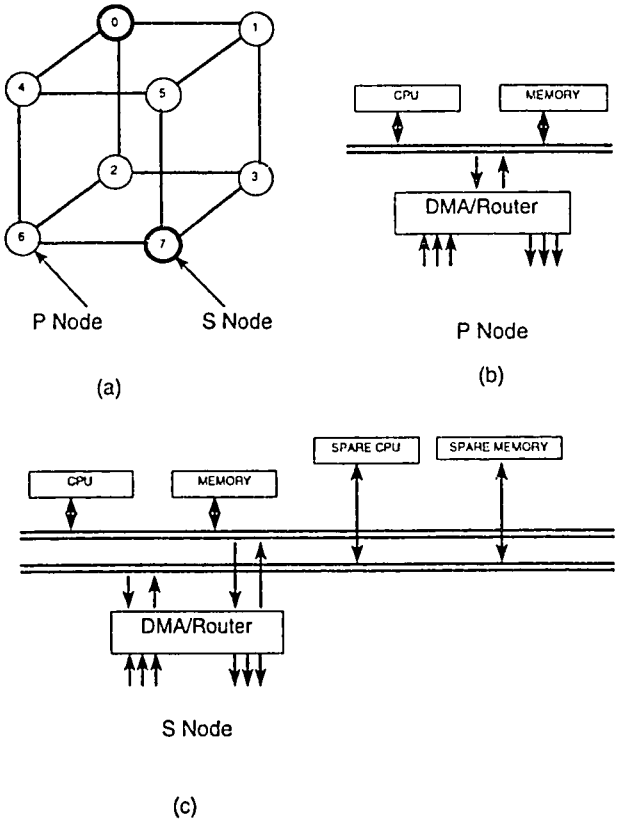

Figure 2.10: The architecture of P and S nodes.

In the second scheme, spare nodes are introduced into the hypercube by inserting them along links connecting particular pairs of processors. This scheme is called the *link spare scheme*. The algorithm for reconfiguration and allocation of these spare nodes in the hypercube is given in [16]. The reliability of this scheme is much better than that of the *node spare scheme* as each node is now covered by more than one spare.

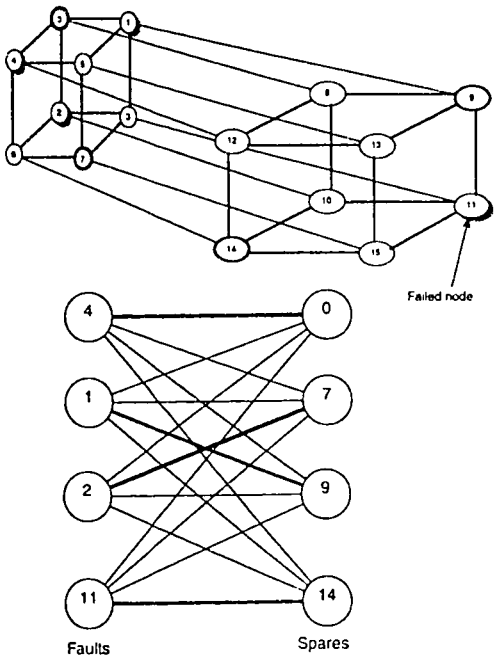

Figure 2.11: Reconfiguration in Banerjee's scheme.

2.6 The 3-FTBB

Hai and Abd-El-Barr [19, 29] have suggested another fault-tolerant hypercube architecture. In this architecture, the basic block can be a 3−cube, in which a spare node has been embedded. This spare node is connected to all the eight primary nodes through spare links. This cube is called a 3-FTBB. This architecture tolerates node as well as link failures. Figure 2.12 shows the cube, and how it tolerates node and link failures. In Figure 2.12(a) we have the original hypercube. In Figure 2.12(b) node 4 has failed and been replaced by the spare node. In Figure 2.12(c), link (1,3) has failed and has been replaced by the links ((1,s),(s,3)).

Subcube reliabilities of this architecture have been analyzed under the following models:

- Node Failure Model
- Link Failure Model

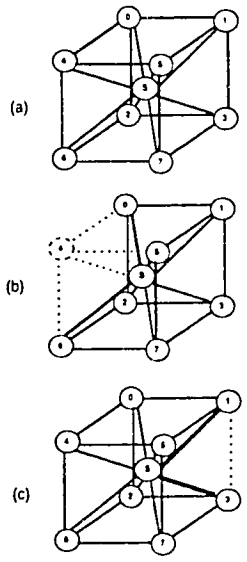

Figure 2.12: Hai and El-Barr's 3-FTBB.

- Combined Node and Link failure model
- Supernode Failure Model

The reliabilities and the MTTF of these basic blocks have been shown to be superior to the ordinary cubes of the same order [29]. In addition, they have analyzed the subcube reliabilities of larger cubes built from these blocks, and show it to be superior to similar large cubes built from ordinary cubes.

2.7 Concluding Remarks

In this chapter, we have discussed some existing fault-tolerant hypercube networks. Most of the presented networks tolerate only node failures. They seek to preserve the physical topology of the hypercube in case of node failures. These schemes

appear to have a very high hardware overhead. In the next chapter, we shall give an overview of some existing hierarchical interconnection networks.

Chapter 3

Existing Hierarchical Hypercube-based Networks

The hypercube is a popular static interconnection topology for interconnecting distributed memory computers. It is a very powerful topology in which other topologies such as rings, meshes, trees etc. can be embedded.

However, the hypercube network has the limitation that increasing the number of nodes requires a change in the basic node configuration. This is a very serious drawback which limits the applicability of the hypercube to very large systems. In addition, the number of links in a hypercube network increases exponentially with dimension. Also, there is some locality of communication among nodes of a hypercube which is not exploited for performance gains.

Hierarchical networks provide a means of exploiting the locality of communications that is inherent in multiprocessor systems. They also provide a means of designing networks with low link costs. A Hierarchical interconnection network can be defined as a network with a distinct hierarchy of interconnections. At the lowest

level, we have clusters of individual nodes, with nodes in each cluster connected by a network. This network is called a Level-1 network. At the next level, groups of clusters are connected by a Level-2 network. The topology at each level can be the same or different. If the topologies at all levels is the same, then the network is called a homogenous HIN; otherwise it is called a heterogeneous HIN. There can be any number of levels in a HIN.

In this chapter, we survey some of the hierarchical hypercube networks proposed in the literature. In Section 3.1 we shall review the binary hypercube based two-level HIN proposed by Dandamudi. Section 3.2 reviews the Hierarchical Hypercube. The Tightly Connected HIN and the Hierarchical Cubic Network are covered in Section 3.3 and Section 3.4 respectively. The Extended Hypercube is explained in Section 3.5. Section 3.6 covers the deBruijn Cube. The General HIN and the FTBB based fault-tolerant HIN are studied in Section 3.7 and Section 3.8 respectively. Section 3.9 concludes this chapter.

3.1 Binary Hypercube based Two-Level Network

Dandamudi [8, 30] proposed a hierarchical network based on binary hypercubes. Figure 3.1 shows a homogenous two level Binary hypercube/Binary hypercube (BH/BH) HIN. Each level one network is called a cluster. A cluster size of eight and two levels have been found to be most optimal under a cost/performance trade-off for connecting very large number of processors in a BH/BH network. Major disadvantages of HIN's include the potential for high traffic on inter-cluster links, and thus the potential degradation in performance, and the potential for diminished fault tolerance due to the special role played by the interface nodes.

The application of standard fault tolerance techniques to HINs can improve their reliability. Dandamudi proposed two techniques using hardware redundancy to improve the fault tolerance of the BH/BH network. In one technique, the level two network is duplicated. In the other technique, a standby spare interface node is provided to reduce the impact of interface node failures on reliability. These two techniques are described below

1. Replication Technique: This technique aims to avoid excessive traffic on inter-cluster links by duplicating the level-two network and having two interface nodes per cluster. This network, shown in Figure 3.2 is referred to as the BH/BH-RS. It is preferred to keep the two interface nodes belonging to a given cluster as far apart as possible in the cluster.

2. Standby Spare Interface Node Technique: In the BH/BH network, the failure of an interface node disconnects the cluster from the network. This can be avoided by providing a standby spare for each interface node in the network. In the event of failure of an interface node, the spare replaces it. This technique is referred to as BH/BH-SI. The BH/BH-RS, with a duplicate level-two network uses more links than the BH/BH-SI, but the BH/BH-SI requires fault detection of the interface node and the extra spare nodes. Dandamudi [8] showed that the performance of the BH/BH-RS is substantially better than the BH/BH-SI which justifies the increase in cost.

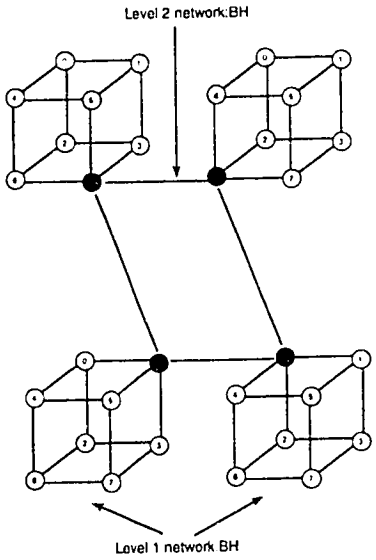

Figure 3.1: A BH/BH HIN with a cluster size of 8.

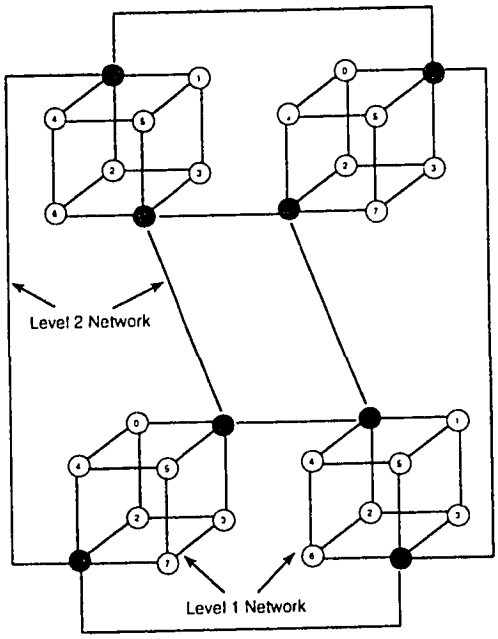

Figure 3.2: A BH/BH-RS HIN with a cluster size of 8.

3.2 The Hierarchical Hypercube

Qutaibah and Bayoumi [22] introduced a new interconnection topology called the Hierarchical hypercube. The structure of an n-HHC (a HHC of order n) consists of three levels of hierarchy. To simplify the description of HHC structure, assume that $n = 2^m + m$ for a non-negative integer m. At the lowest level of hierarchy, we have a pool of 2^n nodes. These nodes are grouped into clusters of 2^m nodes each, and the nodes in each cluster are interconnected to form an $m-$Cube called the Son-Cube or the SCube. The set of SCubes constitutes the second level of hierarchy. A father-cube called the FCube connects the $2^{n-m} = 2^{2^m}$ SCubes in a hypercube fashion. Edges of the SCube are called internal edges and the edges of the FCube are called external edges. An SCube having 2^m nodes is connected to exactly 2^m external edges, and each is incident to one node of the SCube. They generalize this for cases where $n \leq 2^m + m$. In this case, some SCubes won't have external links. Figure 3.3 shows a 5-HHC, where cluster size $m = 2$.

This scheme has a lot of advantages over the conventional hypercube. The degree of each node is much less than the normal hypercube. Also, the Diameter, D of the HHC is bounded by 2^{m+1}. This means that the HHC has Logarithmic diameter. This architecture is scalable. Also, it is possible to embed an $n-$HHC into an $n-$HC. The authors have described some algorithms for Data Communications in the $n-$HHC. These algorithms can be used to improve the fault-tolerance of the $n-$HC also. The fault-tolerance aspects of the HHC have not been considered by the authors [22].

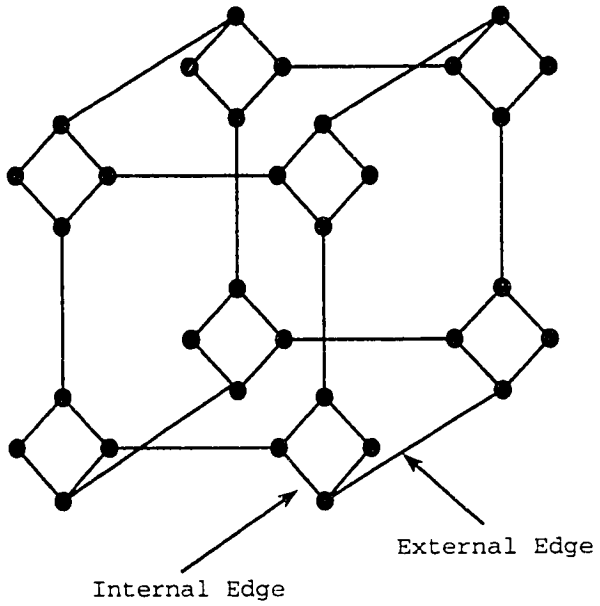

Figure 3.3: A 5-HHC.

3.3 The Tightly Connected Interconnection Network

Breznay and Lopez [31] proposed a hierarchical interconnection network where isomorphic clusters are connected using a complete graph at the highest level of hierarchy. The authors proposed a HIN based on the concept of a *tightly connected interconnection network* or TCN. In a TCN, every node serves as the interface, thereby distributing the communication load evenly throughout the system. This is accomplished by using a complete graph as a meta-network, the network which connects the clusters.

Figure 3.4 shows a static singly linked tightly connected hierarchical interconnection network (1TCN). It consists of 2-cube cluster's which are interconnected

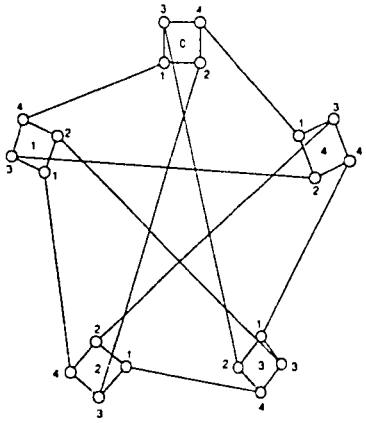

Figure 3.4: A 2-Dimensional hypercube based 1TCN.

by a fully connected graph. If the cluster consists of n nodes, then there will be $n+1$ clusters in the 1TCN. A routing algorithm is developed for this HIN which is near-optimal, i.e. for up to more than one million nodes, this algorithm finds the true diameter and the average distance is within 5% of the optimal average distance.

3.4 The Hierarchical Cubic Network

Ghose and Desai [28] proposed an interconnection network - Hierarchical Cubic Network, HCN, using hypercube network as the basic building block. The HCN(m,n) has 2^m clusters, where each cluster is a hypercube network of order n, and $m \leq n$. When $m = n$ it is called a *complete* HCN, when $m < n$ it is called an *incomplete* HCN. A HCN(2,2) is shown in Figure 3.5.

Each processing element (PE) in a cluster has $n+1$ connections, of which n are *local links* corresponding to the n-cube connection, and the additional link - called the *external* link is used to connect to a PE in some other cluster. In an incomplete

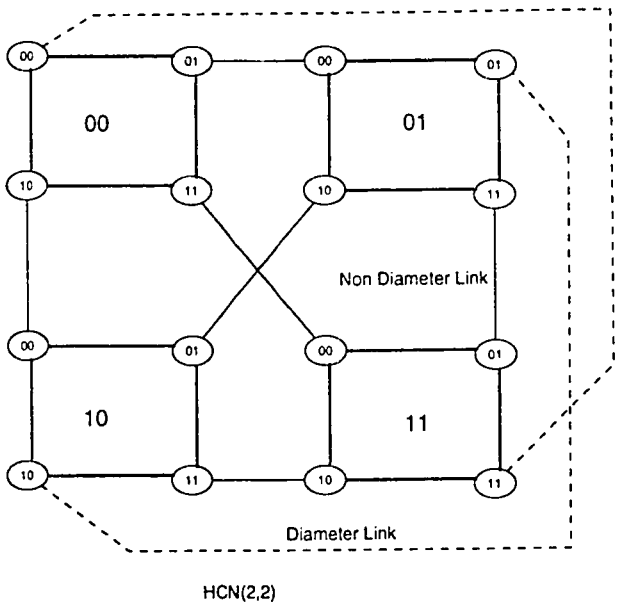

Figure 3.5: A Hierarchical Cubic Network with m=2 and n=2.

HCN, some external links may be unused. Routing algorithms have been proposed for the HCN, and the message path length in the HCN under these is shown to be bounded by $2n$.

The HCN has many advantages. The diameter $D \leq m+n$ and degree $d = n+1$ for a HCN(m,n). It has a lower diameter than a comparable hypercube, and requires half the number of connections per node than the hypercube. It is possible to increase the size of an incomplete HCN(m,n) by adding more clusters, as long as $m \leq n$. However, this also means that the HCN is not expandable beyond this point, unless the node degree is increased.

3.5 The Extended Hypercube

In [23], Kumar and Patnaik proposed the Extended Hypercube (EH). This is a hierarchical, expansive, recursive structure with a constant predefined building block. The basic module of an EH- $EH(k,1)$ consists of a k-cube of processor elements (PE's) and a Network Controller (NC) as shown in Figure 3.6. The PE's are connected to the NC via links, thus the degree of each PE is $k+1$. The NC is used as a communication processor to handle intermodule communication. The basic module is said to be an EH of degree one. It is a constant predefined building block and the node configuration remains the same regardless of the degree of the EH.

The EH architecture can easily be extended by connecting the basic modules. We can connect eight EH(3,1) (basic modules) to get an EH(3,2) as shown in Figure 3.6. The interconnection topology formed by the 3-cube of NC's at the first level and a controller at the second level is identical to that of the basic module. Thus we have a hierarchical structure consisting of 64 PE's at the zero'th level (lowest level), eight NC's at the first level and one NC at the second level.

The authors have compared the EH(k,l) with a hypercube having 2^{k+l} nodes, considering only the PE's of the EH. The diameter of the EH is given by $[k+2*(l-1)]$, whereas the diameter of the hypercube with an identical number of nodes is $k \times l$. For $k > 2$ and $l > 1$, the diameter of the EH is always lesser than that of the n-cube $(n = k * l)$. The diameter of the EH is less than that of the hypercube, the HCN, and the cube connected cycles, but more than that of the pyramid. The degree or connectivity of the EH is constant for a given k. This means that hardware changes in a node are not required for extending the architecture. In contrast, the connectivity of a hypercube is proportional to the dimension of the cube. However

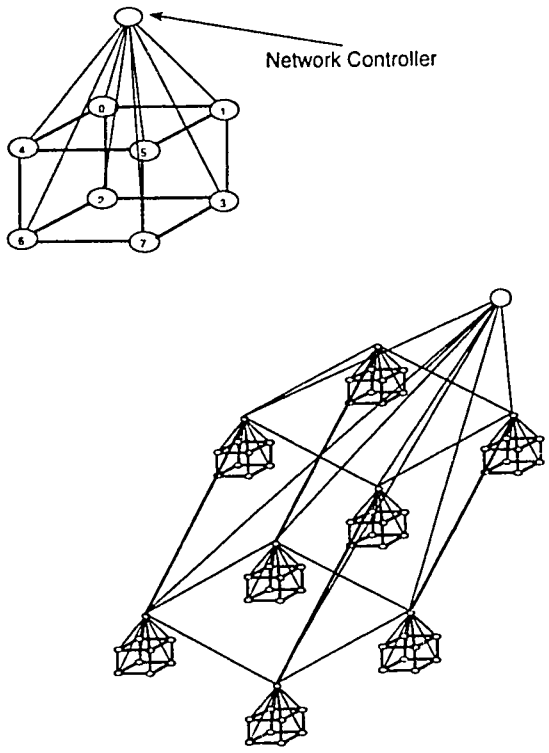

Figure 3.6: An EH(3,1) and a EH(3,2) constructed from 8 EH(3,1)s.

the mean internodal distance of the EH is not as small as that of other networks. The EH has poor fault-tolerance for communication from one level to another, as the nodes of an EH(k,l), for $(0 \leq j \leq l-1)$ rely only on one communication processor for interaction with the NC's at higher levels. Another disadvantage of this architecture is the high degree of the Network Controller.

3.6 The dBCube

A class of hierarchical networks suitable for VLSI implementation were introduced by Chen and Agrawal in [32]. These networks, called the dBCube, use the hypercube as the basic cluster, and connect the clusters using a deBruijn graph. The dBCube is a compound graph of a deBruijn graph and a hypercube obtained by replacing each node of deBruijn graph with a hypercube cluster. It can then utilize the topological properties of the hypercube for easy embedding of existing parallel algorithms, and the short diameter of the deBruijn graphs to provide quick intercluster communications. Regularity and expandability are two main features of deBruijn graphs.

In a dBCube, a *local link* connects two nodes in the same cluster, and a *remote link* connects two nodes in different clusters. The connection topology for each cluster is a hypercube, while the topology for connecting all clusters is a deBruijn graph. The deBruijn graph is well studied in [25, 33, 34]. It's construction preserves the following properties.

- For a given degree it has a small diameter (smallest in most cases) with respect to the number of nodes in the graph.

- It is possible to find a distributed routing control, which can be implemented locally at each node. Fault-tolerant routing can degrade performance gracefully.
- The number of nodes in a deBruijn graph is not limited to a power of two. Also, when generalized, it can consist of an arbitrary number of nodes, including prime numbers.

To construct a dBCube, each node in the deBruijn graph is replaced by a hypercube cluster, and the deBruijn graph is used to connect different clusters. There is one remote link associated with each node of a cluster. The degree of each node in the resultant network is the same for every node. Thus, the dBCube can be extended to larger networks, while keeping the node architecture the same. A dBCube is characterized by $dBC(c,d)$ where c is the number of cubes/cluster (or the size of the deBruijn graph) and d is the dimension of the cube. The total number of nodes in a $dBC(c,d)$ is $c \times 2^d$. A dBCube(8,2) is shown in Figure 3.7.

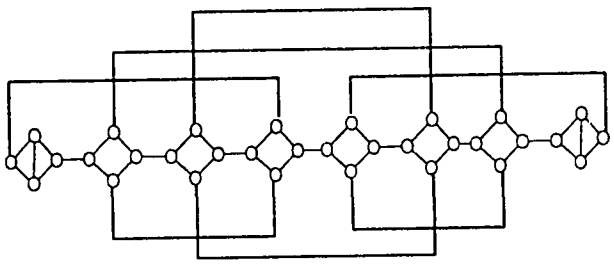

Figure 3.7: A dBCube(8,2) where the size of the dBGraph is 8 and each module is a hypercube of dimension 2.

The authors have evaluated the dBCube structure using diameter, degree and cost metrics. Compared to the other compound hierarchical networks Cube Connected Cycles (CCC) and HCN, the dBCube exhibits much lower degree and cost

than the HCN, much smaller diameter than the CCC, and eventually becomes more cost effective than the CCC for large networks. The dBCube can be expanded by changing the size of the level-two network, which is a deBruijn graph. A deBruijn graph can be expanded by adding as many nodes as needed. This means that the minimum number of nodes to be added to the dBCube for expansion is very small, and equals the size of the basic cluster.

3.7 General Hierarchical Interconnection Networks

In [35] a class of general hierarchical interconnection networks for message passing is proposed. In these networks, more than one node from each cluster can act as interface nodes. The procedure for construction is as follows. The total number of nodes N in a hierarchical network are divided into K_1 clusters of N/K_1 nodes each. Each cluster of N/K_1 nodes is connected to form a level-1 network. The nodes in every cluster are ordered in the same way, i.e., the corresponding nodes in different clusters have the same internal address. Then I_1 nodes, $1 \leq I_1 \leq N/K_1$, from each cluster are selected to act as interface nodes. To construct level-2 networks, these interface nodes are first divided into I_1 groups. Each group consists of K_1 nodes, which are from K_1 different clusters, i.e. one node from each cluster with the same internal address. Then, the K_1 nodes of each group are again divided into K_2 clusters of K_1/K_2 nodes each. Each cluster is connected to form a level-2 network, and I_2 nodes, $1 \leq I_2 \leq K_1/K_2$, are selected as the interface nodes to construct the level-3 networks and so on. The interconnection networks used to construct clusters at

different levels may have the same or different topologies. The performance analysis is done for hierarchical networks in which the second level network is a hypercube, and the first level network either a Completely Connected or hypercube topology. These are referred to as BH/BH and CC/BH respectively. Figure 3.8 shows a

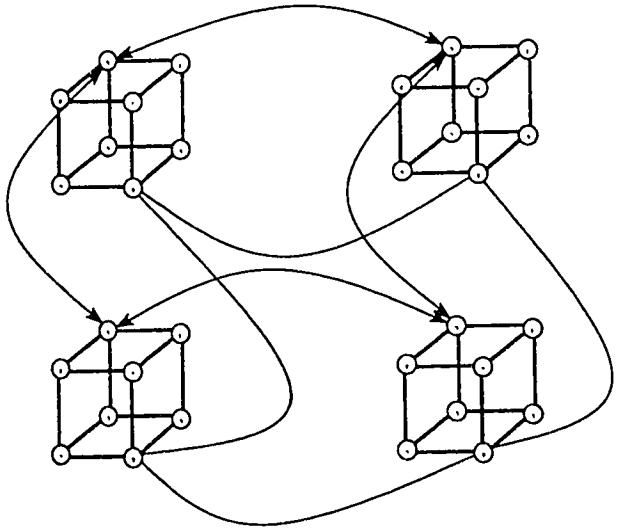

Figure 3.8: A hierarchical interconnection network based on hypercubes.

3-cube/3-cube hierarchical interconnection network. The measures considered are diameter, mean internode distance, traffic density, and total number of links. A methodology is given to compute the various parameters like number of clusters, and the number of interface nodes which optimize a given performance measure.

3.8 Hierarchical Network based on FTBBs

Hai [19] proposed a hierarchical network based on his fault-tolerant basic block (FTBB). As discussed in Chapter 2, his FTBB is a binary hypercube of dimension 'd' to which a spare node has been added for fault-tolerance. This spare node is

connected to all the nodes of the hypercube. The hierarchical network proposed is a modification of Dandamudi's BH/BH-RS proposal [8], where the Binary Hypercube based clusters have been replaced by 3-FTBBs. This scheme is called BH/BH-FTBB-RS scheme. In this scheme, the level-two network is replicated on the spare nodes. This is shown in Figure 3.9. Subcube reliability is calculated for this scheme.

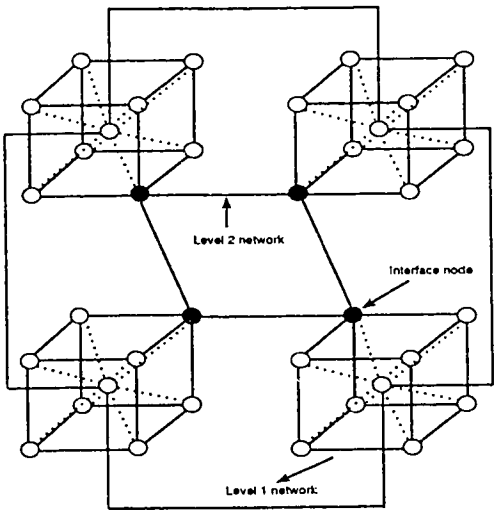

Figure 3.9: A two-level hierarchical network using Hai's FTBB as the basic block.

It is shown that the subcube reliability of this scheme is substantially better than that of the BH/BH-RS or the BH/BH-SI scheme of Dandamudi.

3.9 Concluding Remarks

This chapter presented an overview of some of the hierarchical, hypercube-based interconnection networks proposed in the literature. Of these schemes, only Hai's [19, 29] and Dandamudi's [8] investigated the aspect of fault-tolerance. Networks which are based on trees like the EH [23] have a high message density towards the

root. Some of these networks are not easily expandable. Networks based on the deBruijn graph are irregular, and it is difficult to map algorithms to them. In the next chapter, we shall come up with the design for a hierarchical interconnection network.

Chapter 4

Design and Analysis of a Hierarchical Interconnection Network

Interconnection networks are critical elements of multiprocessor systems. Their design influences many performance parameters like granularity of parallelism, expandability, scalability, fault-tolerance, etc. The properties of interconnection networks can be investigated by modelling them as graphs. Different interconnection networks have different properties. For example, ring networks are easily expandable but don't scale very well. The binary tree has a diameter and mean internodal distance logarithmic in the number of nodes, but message density increases rapidly towards the root. The hypercube has low diameter and mean internodal distance, but does not expand very well. The mesh and torus networks are expandable, but their diameter increases linearly with the dimensions. Also, the link cost for interconnection networks, measured as the total number of links used in the network

[8], increases drastically as the network size increases. This is particularly true for structures like hypercubes. Thus, it can be seen that no network has all the desirable characteristics, they usually offer a tradeoff between these characteristics.

In order to get good performance, it is essential to match the structure of the problem to the topology of the underlying interconnection network. Usually, there is some locality of reference among the tasks of a parallel application. This can be exploited to achieve performance improvement. For example, tasks which have a high degree of interaction could be grouped together and mapped to processors which are densely connected with each other. In this portion of the network, there might be multiple paths between source and destination nodes. For realizing interconnection between the groups, a sparsely connected topology could be chosen, as locality of reference implies very little communication among these groups.

Hierarchical interconnection networks provide a framework whereby different interconnection topologies could be integrated in order to incorporate the desirable features of each into a single network. They are also suited to take advantage of the locality of communications that exists among tasks of a parallel application.

Hierarchical interconnection networks consist of multiple levels. The level-one network, also called basic block, cluster or FTBB, contains most of the processing elements. The algorithms running on these nodes should be able to communicate with each other efficiently. Interconnection networks which provide very low communication delays are desirable at this level. The level-two network provides communication between the blocks or clusters. It should have the potential for carrying high traffic.

In this chapter, we investigate the performance of hierarchical interconnection networks constructed using different basic building blocks. Our objective is to come

up with a design that is expandable and has lesser number of links as compared to it's component networks. Expandability implies that the node complexity should remain constant and not change with network size.

The layout of this chapter is as follows. In Section 4.1 we investigate the design of the basic block. Different architectures are considered for this. The design of the level-two network for the proposed hierarchical architecture is investigated in Section 4.2 . This is followed by the performance analysis of the various hierarchical architectures. In Section 4.3 we investigate the performance of a three-dimensional binary hypercube based architecture. This is followed by the performance of the Folded cube based architecture in Section 4.4. The performance of a Möbius cube based architecture is given in Section 4.5. This is followed by Section 4.6 which gives a discussion on the results obtained. The properties of the architectures are investigated in detail in Section 4.7. Section 4.8 concludes this chapter.

4.1 Design of the Basic Block

The basic blocks contain the processing elements, which communicate with each other. Locality of reference implies that there will be more communication among processors in the same cluster as compared to that among processors in different clusters. Therefore, the basic block topology should have low mean internodal distance in order to have good performance as compared to non-hierarchical networks. The hypercube is one topology which has low mean internodal distance and diameter. In the following, we investigate the hypercube and it's variants as possible cluster network topologies.

The hypercube architecture and it's variations have become popular due to

it's powerful interconnection features. Many algorithms can be executed on these, either through direct mapping or through subnetwork embedding. Some of these, like the Möbius cube [36] and the Folded Hypercube [37] have lesser cost than the hypercube.

4.2 Design of the Level-Two Network

The mesh, tree and torus are three architectures which have uniform degree for the nodes. The tree has a disadvantage that a node failure partitions the network. Mesh networks are not uniform, as nodes on the edge have different degrees than nodes in the middle. We investigate the torus as a topology for the level-two network. Consider a torus of width W. It has $N = W^D$ nodes and DW^D links where D is the dimension of the torus. It has an mean internodal distance of $DW/4$. Let us see how scalability and message delay are related in a torus. A 2D torus of width W has $N = W^2$ nodes. The network size N can be increased by either:

1. Keeping dimension D fixed and increasing W as $N^{1/D}$, or
2. Keeping W fixed and increasing dimension D as $\log N$.

Since one of our objectives is to have uniform degree for all nodes, the first option given above is preferable. However, the message path-length increases relatively rapidly as $\sqrt[D]{N}$.

Under the assumption of locality of reference, we can be reasonably sure that this growth of mean internodal distance will not affect the network adversely. This is particularly true if the cluster network topologies have low message path-length

growth. Thus, our level-two network will be a torus which will interconnect basic-blocks at level-one. This is shown in Figure 4.1. The level-two network is duplicated so that all the nodes in a module or cluster become interface nodes. Also, this decreases the mean internodal distance and diameter of the architecture.

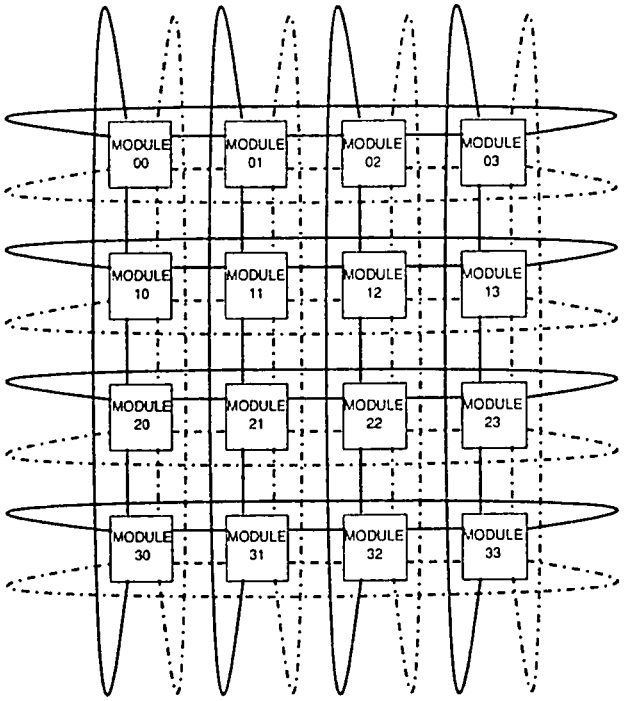

Figure 4.1: A hierarchical interconnection network based on the torus, using hypercube based clusters.

4.3 *HyperTorus*

The first alternative which we examine for the basic-block is the hypercube. It has been shown in [8] that a cluster size of eight is optimal for a hierarchical network built using hypercubes. Therefore, we consider using a 3-cube as a basic block in the hierarchical network. We propose to use all nodes as interface nodes, thus having a

uniform degree for all nodes. The level-two torus network is duplicated to achieve better fault-tolerance and also decrease mean internodal distance.

Thus, the proposed scheme is a two-level hierarchical network. At the lowest level, we have a 3-cube as the basic block. These blocks are interconnected in a 2D-torus topology at the next level. If the size of the torus is $M \times N$ (M Rows and N Columns), then we have $M \cdot N$ blocks in the architecture. Each node in the torus at level two is a 3-cube. The basic block of our scheme is shown in Figure 4.2. The solid lines connect to one torus network and the dashed lines connect to another torus. This architecture is called the *HyperTorus*.

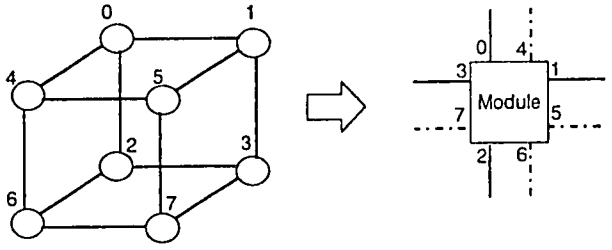

Figure 4.2: Hypercube based proposed basic building block.

We compute the LP product and mean internodal distance of this architecture for a range of network sizes. This is shown in Table 4.1. The cost of this architecture is much less than that of a hypercube architecture having the same number of nodes. For networks of size 1024 nodes, the cost of the hypercube and the *HyperTorus* are equal. This is because the hypercube has a logarithmic diameter which offsets the disadvantage due to high node degree as the network size increases.

Table 4.1: Performance measures of the *HyperTorus* network.

M	N	Size	Links	Diameter	M.I.D	Cost	$L \cdot P$
2	2	32	64	6	3.0	24	192
2	4	64	128	8	4.03	32	516
4	4	128	256	9	5.09	36	1304
4	8	256	512	13	7.125	52	3648
8	8	512	1024	17	9.164	68	9384
8	16	1024	2048	25	13.18	100	27000

4.4 Folded Hypercube based Architecture

The Folded hypercube [37] augments the hypercube with extra links connecting the farthest nodes. For example, a Folded hypercube of dimension d is the same as a normal hypercube of the same dimension, but having 2^{d-1} extra links. Each link connects the two farthest nodes in the hypercube, i.e., those nodes at a distance d apart. This reduces the diameter of the hypercube. A Folded hypercube of dimension three is shown in Figure 4.3. We use a 3-dimensional Folded hypercube

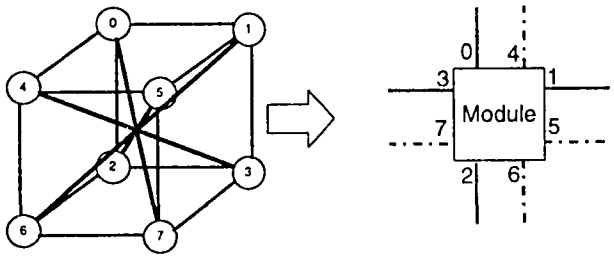

Figure 4.3: A Folded hypercube of order three.

as a cluster for the hierarchical interconnection network. This network is denoted as the Torus/Folded-cube network. The performance of this network is shown in Table 4.2. When compared with the previous proposal, it can be seen that the

Table 4.2: Performance measures of the Torus/Folded-cube network.

M	N	Size	Links	Diameter	M.I.D	Cost	$L \cdot P$
2	2	32	80	5	2.62	25	210
2	4	64	160	7	3.62	35	580
4	4	128	320	9	4.60	45	1470
4	8	256	640	13	6.57	65	4210
8	8	512	1280	17	8.55	85	10950
8	16	1024	2560	25	12.54	125	32110

additional number of links have increased both the cost and the LP product of the network. Due to the extra links, the diameter is the same as that of the *HyperTorus* for network sizes up to 128 nodes. But after that, it increases drastically.

4.5 Möbius Cube based Architecture

In [36], a variant of the hypercube architecture, called the Möbius cube is given. It gives better performance as compared to the binary hypercube in terms of cost and mean internodal distance, while using the same number of nodes and links . The Möbius cube is a generalization of the twisted N-cube architecture.

A Möbius cube of dimension n has 2^n nodes. Each node has a unique n-component binary vector for an address. Consider a node **X** having an address $(X_1, X_2, ..., X_n)$. Each node **X** connects to n neighbors $Y_1, Y_2, ..., Y_n$ where each Y_i satisfies one of the following equations.

$$Y_i = (X_1..., \bar{X}_i X_{i+1}...X_n) \; if \; X_{i-1} = 0 \tag{4.1}$$

$$Y_i = (X_1..., \bar{X}_i \bar{X}_{i+1}...\bar{X}_n) \; if \; X_{i-1} = 1 \tag{4.2}$$

The connection between **X** and **Y** along dimension 1 has X_0 undefined, so we can either assume $X_0 = 0$ or $X_0 = 1$. With the former, we get 0-Möbius cubes and with the latter, we get 1-Möbius cubes. A Möbius cube of dimension three is shown in Figure 4.4.

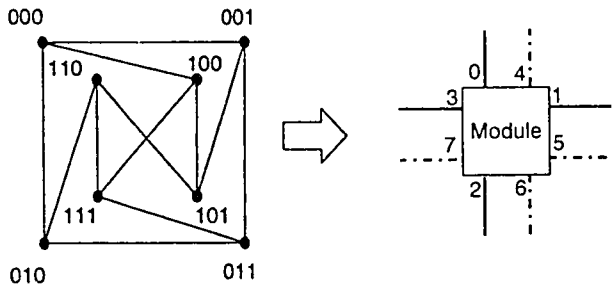

Figure 4.4: A Möbius cube of dimension three.

We examine the use of a Möbius cube of dimension three as the basic block in the hierarchical network. This network is denoted as the Torus/Möbius-cube network. In Table 4.3, we show the performance of the Torus/Möbius-cube network. When compared with the previous proposals, we can see that the *LP* product of this network is smaller than that of the Torus/Folded-cube, and comparable to that of the *HyperTorus*. This is due to the fact that the basic block has a smaller diameter and mean internodal distance, as compared to the 3-cube. One disadvantage of the Möbius cube is that embeddings of the hypercube have a dilation of three. In other words, if a hypercube algorithm were mapped to a Möbius cube, tasks which were running on adjacent nodes in the hypercube may not be on adjacent nodes in the Möbius cube. In the presence of faults, this dilation will increase further.

Table 4.3: Performance measures of a hierarchical network with a 3-dimensional Möbius cube as the basic block.

M	N	Size	Links	Diameter	M.I.D	Cost	$L \cdot P$
2	2	32	64	5	20	2.75	176
2	4	64	128	7	28	3.88	497
4	4	128	256	9	36	4.88	1250
4	8	256	512	14	56	7.05	3614
8	8	512	1024	17	68	9.06	9286
8	16	1024	2048	26	104	13.17	26991

4.6 Results and Discussion

In this section, we discuss the performance of the various architectures given above. We use *LP* Product [8] and cost (*degree* × *diameter*) as a metric for comparison. In Figure 4.5, we plot the costs of various architectures for network sizes ranging from 32 to 1024 nodes. The architectures considered are the *HyperTorus*, the Torus/Möbius-cube and the Torus/Folded-cube networks.

Table 4.4 tabulates the diameter and cost of the three architectures considered. We see that the Torus/Folded-cube network has a lesser diameter than the *HyperTorus* for network sizes ranging from 32 to 64 nodes. For larger networks, their diameters are the same. However, the Torus/Folded-cube has a higher degree on account of the extra links. Therefore, it's cost is more than that of the *HyperTorus*as shown in Figure 4.5. The Torus/Möbius-cube also has a lesser diameter than that of the *HyperTorus* for network sizes ranging from 32 to 64 nodes. For other sizes, their diameter is almost the same. This is because the irregular structure of the Möbius cube offsets it's advantage of lesser diameter as compared to the hypercube.

Table 4.5 and Figure 4.6 shows the relation between network size and *LP* prod-

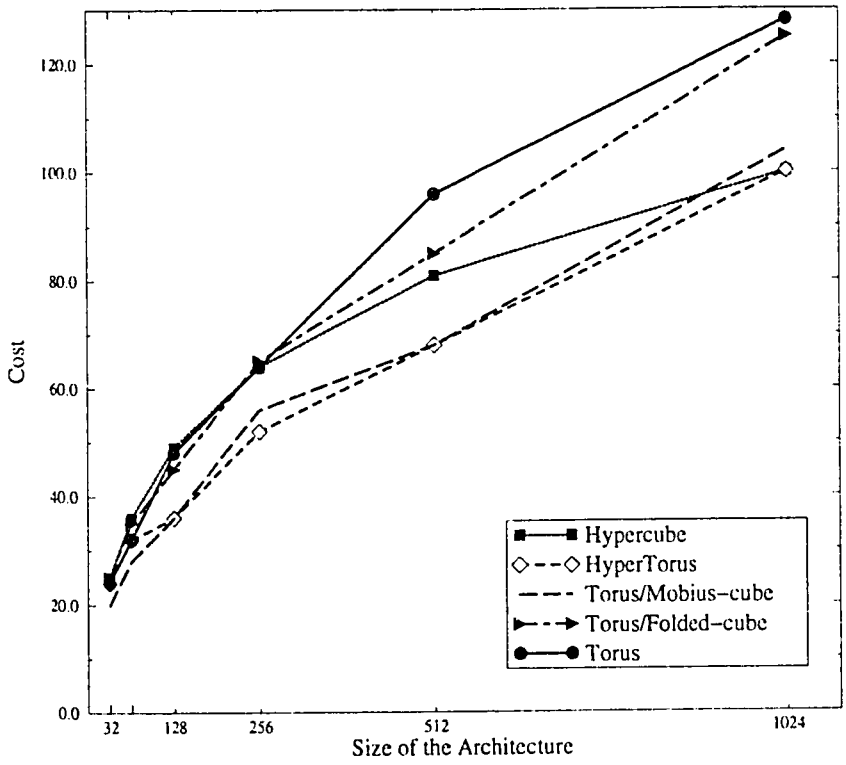

Figure 4.5: Costs of the *HyperTorus*, Torus/Folded-cube and the Torus/Möbius-cube networks.

Table 4.4: Diameters of the *HyperTorus*, the Torus/Folded-cube and the Torus/Möbius-cube networks.

Size	*HyperTorus*		Torus/Folded-cube		Torus/Möbius-cube	
	Diameter	Cost	Diameter	Cost	Diameter	Cost
32	6	24	5	20	5	20
64	8	32	7	28	7	28
128	9	36	9	36	9	36
256	13	52	13	52	14	56
512	17	68	17	68	17	68
1024	25	100	25	100	26	104

Table 4.5: Mean internodal distances (M.I.D) and *LP* products of the *HyperTorus*, the Torus/Folded-cube and the Torus/Möbius-cube networks.

Size	*HyperTorus*		Torus/Folded-cube		Torus/Möbius-cube	
	M.I.D	*LP*	M.I.D	*LP*	M.I.D	*LP*
32	3.0	192	2.62	210	2.75	176
64	4.03	516	3.62	580	3.88	497
128	5.09	1304	4.60	1470	4.88	1250
256	7.125	3648	6.57	4210	7.05	3614
512	9.164	9384	8.55	10950	9.06	9286
1024	13.18	27000	12.54	32110	13.17	26991

uct for these architectures. Again, the low mean internodal distance advantage of the Torus/Folded-cube (see Table 4.5) is offset by the extra links in the architecture. Therefore, it has a higher *LP* product than the other networks. The *HyperTorus* and the Torus/Möbius-cube networks have a lesser *LP* product than that of the hypercube for small network sizes. For large network sizes (> 1024 nodes), the hypercube has a lower *LP* product than the *HyperTorus*. This is because the hypercube has a lower mean internodal distance ($= (\log N)/2$), which offsets it's disadvantage due to more links. The Torus/Möbius-cube network has a marginally lower *LP* product than the *HyperTorus*. From Figure 4.6, we can see that this difference is very small, and the *LP* products of both the architectures are about the same.

Thus, the *HyperTorus* architecture is a feasible alternative to the Torus/Folded-cube and the Torus/Möbius-cube networks.

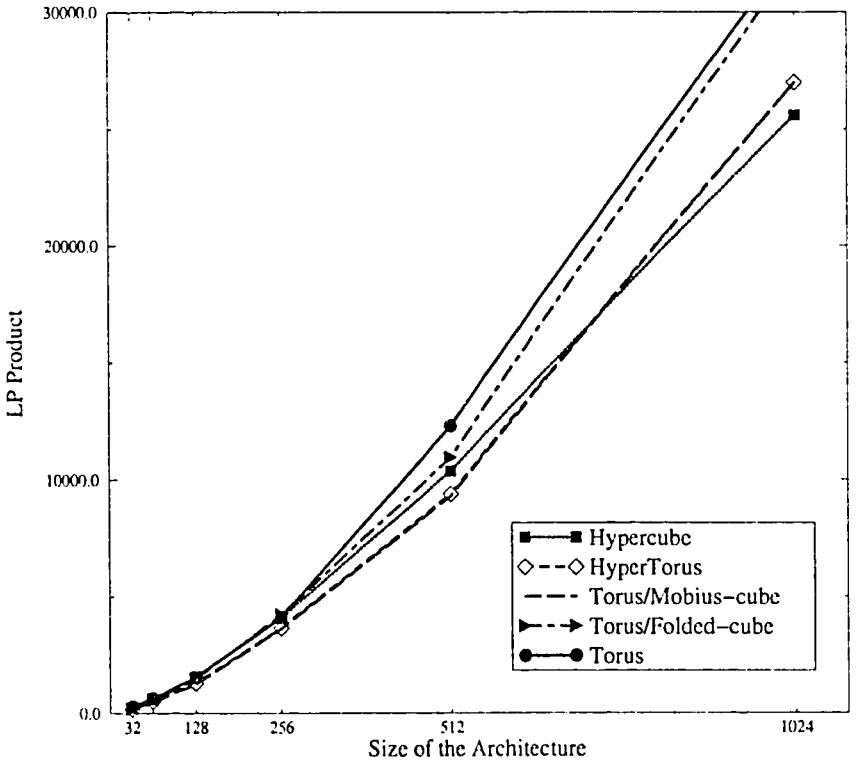

Figure 4.6: *LP* products of the *HyperTorus*, the Torus/Folded-cube, and the Torus/Möbius-cubenetworks.

4.7 Properties of the Proposed Networks

In this section we discuss the properties of the proposed architectures. First, we determine the properties of the *HyperTorus* architecture. The properties of the Torus/Möbius-cube and the Torus/Folded-cube are similarly determined. A comparison is made between the *HyperTorus* and other recursive and hierarchical architectures.

The basic block of Figure 4.2 consists of a 3-cube. A torus network at level two connects the basic blocks. There are $M \cdot N$ basic blocks in a $M \times N$ architecture. Each basic block has 8 nodes. There is a total of $8 \cdot M \cdot N$ nodes in the architecture. Each basic block has 12 links. Thus, the basic blocks contribute $12 \cdot M \cdot N$ links. The level-two network contributes $4 \cdot M \cdot N$ links. Therefore, there are $16 \cdot M \cdot N$ links in the architecture. Each node in the architecture is connected to three neighbors in the basic block to which it belongs and one neighbor in another basic block. Thus the degree of each node is four.

Theorem 1 *The diameter of a* $M \times N$ HyperTorus *architecture is bounded by* $2(\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor) + 4$

Proof: Consider a $M \times N$ architecture. Since the level-two network is a torus, we have the diameter of the torus as $\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor$. Now consider two farthest modules in the architecture, for example $(0,0)$ and $(\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor)$. A typical path between these two would be $(0,0) \rightarrow (0,1) \rightarrow \ldots (0, \lfloor \frac{N}{2} \rfloor) \rightarrow (1, \lfloor \frac{N}{2} \rfloor), \rightarrow (2, \lfloor \frac{N}{2} \rfloor) \ldots (\lfloor \frac{M}{2} \rfloor, \lfloor \frac{N}{2} \rfloor$. The path length in each intermediate block is one, when the direction of traversal is horizontal or vertical. There are $\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor - 2$ such blocks. The maximum path length in the module $(0, \lfloor \frac{N}{2} \rfloor)$ is three, as there is a change of direction (from

Table 4.6: Topological comparison of hierarchical networks based on the Torus and Hypercube.

	HyperTorus	Torus/Folded-cube	Torus/Möbius-cube
Size	$8 \times M \times N$	$8 \times M \times N$	$8 \times M \times N$
Degree	4	4	4
Diameter	$2(\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor) + 4$	$2(\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor) + 2$	$3(\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor)$
Cost	$4(2(\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor) + 4)$	$4(2(\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor) + 2)$	$12(\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor)$

Table 4.7: Topological comparison of recursive networks.

Network	Size	Degree	Diameter	Cost
BHC	2^n	n	n	n^2
FH	2^n	$n+1$	$\lceil \frac{n}{2} \rceil$	$n(n+1)$
Möbius	2^n	n	$\lceil \frac{n+1}{2} \rceil$	$n(\lceil \frac{n+1}{2} \rceil)$
CBT	$2^n - 1$	3	2(n-1)	6(n-1)
NNM	r^n	2n	$n\lfloor \frac{r}{2} \rfloor$	$2n^2\lfloor \frac{r}{2} \rfloor$
GHC	r^n	n(r-1)	n	$n^2(r-1)$
HyperTorus	$8 \times M \times N$	4	$2(\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor) + 4$	$4(2(\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor) + 4)$

horizontal to vertical or vice versa) in the path. Similarly, the maximum path length in the destination module ($\lfloor \frac{M}{2} \rfloor, \lfloor \frac{N}{2} \rfloor$ is three. Adding all of these, we get an upper bound on the diameter of the architecture as $2(\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor) + 4$. ■

A similar reasoning is used to derive the properties of the Torus/Möbius-cube and the Torus/Folded-cube networks. The Torus/Folded-cube network has more links than the *HyperTorus* and the Torus/Möbius-cube networks. There are $20 \cdot M \cdot N$ links in the Torus/Folded-cube architecture. The characteristics of the *HyperTorus*, the Torus/Folded-cube, and the Torus/Möbius-cubeare summarized in Table 4.6.

In Table 4.7, we compare the topological characteristics of some existing recursive networks with the topological characteristics of the *HyperTorus*. As can be

Table 4.8: Topological comparison of hierarchical networks.

Network	Size	Degree	Diameter	Cost
CCC	$d \times 2^d$	3	$\lfloor \frac{5d-2}{2} \rfloor$	$3 \times \lfloor \frac{5d-2}{2} \rfloor$
HCN	2^{2d}	$d+1$	2d	2d(d+1)
HFN	2^{2d}	$d+2$	$2\lfloor \frac{d}{2} \rfloor + 1$	$(2\lfloor \frac{d}{2} \rfloor + 1)(d+2)$
dBCube	$c \times 2^d$	d+1	d(1+log c)	d(d+1)(1+log c)
HHC	$2^{n=2^m+m}$	$m+1$	2^{m+1}	$(m+1)2^{m+1}$
EH	2^{k-l}	k+1	k+2(l-1)	(k+1)(k+2(l-1))
HyperTorus	$8 \times M \times N$	4	$2(\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor) + 4$	$4(2(\lfloor \frac{M}{2} \rfloor + \lfloor \frac{N}{2} \rfloor) + 4)$

seen from the table, the Binary Hypercube (BHC), Folded Hypercube (FH) and the Möbius cube have cost quadratic in the dimension of the networks. The Complete Binary Tree (CBT) has a constant degree, but low fault-tolerance. Also, there is high message density towards the root. The Nearest Neighbor Mesh (NNM) has a diameter which grows linearly with network size. The Generalized Hypercube (GHC) has the same diameter as the hypercube, but lesser cost. The *HyperTorus* has cost linear in the dimensions of the network.

Table 4.8 compares the topological properties of the *HyperTorus* with those of existing hierarchical networks. The Cube Connected Cycles has a constant degree. The Hierarchical Cubic Network and the Hierarchical Folded Network have a lower diameter and link cost than the hypercube. However, they are not expandable. The Hierarchical Hypercube (HHC) has a logarithmic diameter, and degree lower than the hypercube. However, the HHC is also not expandable beyond a point. The Extended Hypercube (EH) has a constant degree and diameter lesser than that of the hypercube. However, it is tree based, and the Network Controller in the EH is a bottleneck. The *HyperTorus* has a constant degree which gives it the advantage of expandability.

4.8 Concluding Remarks

This chapter investigated the design of a hierarchical multicomputer network. The primary feature which we desired in a hierarchical network was good performance and expandability. Expandability means that node degree should not change if the network size is to be increased. The level-two network was chosen to be a torus, because of it's symmetric nature and constant, low node-degree. Different basic building blocks were evaluated, in terms of their respective performance, for possible incorporation as level-one network. A candidate was chosen as the basic block for the proposed hierarchical network, based on it's cost and LP product. It's properties were analyzed in detail.

In the next chapter, we shall investigate the design of a hierarchical fault-tolerant interconnection network. The basic block for this architecture will be chosen from among various alternatives by evaluating it's performance characteristics. It's fault-tolerance aspects will be studied through simulations. A reconfiguration strategy will be given which will allow the block to recover from faults. The cost/performance tradeoff of the selected block will be evaluated and compared to that of the hypercube and the torus.

Chapter 5

Design and Analysis of a Hierarchical, Fault-Tolerant Network

In the previous chapter, the design of the *HyperTorus*, a hierarchical network based on the hypercube was discussed. It was shown that this architecture performs better than the Torus/Möbius-cube and the Torus/Folded-cube networks. This architecture is expandable, and the node complexity does not change as the network size increases.

As the network size increases, the probability of failure increases. This is because each additional component adds to the probability of failure of the network. Thus it becomes necessary to incorporate fault-tolerance into the network. This can be done by incorporating redundant elements into the system.

Redundancy can be incorporated in two ways, using either global or modular sparing. In global sparing, a spare can replace any failed node in the network. A

scheme based on global sparing was presented in [13]. Global sparing is very costly in terms of the number of links and hardware switches required for reconfiguration. In modular sparing, a spare node can replace only those nodes which are in some sphere of locality. This makes use of the recursive construction property of the architecture to incorporate fault-tolerance.

Fault-tolerance can be incorporated elegantly into hierarchical networks, using the concept of modular sparing. Each module, or cluster of the network is made internally redundant. The spares in a module are used to cover failures in that module. This leads to less hardware overhead and simpler reconfiguration strategies, at the cost of reduced spare utilization.

Modular, fault-tolerant hypercubes described in [12, 13, 14] tolerate only node failures. System reliability analysis for these schemes does not take link failures into account. Link failures cannot be neglected as there are more links than nodes in the hypercube. Most of the fault-tolerant schemes use hardware switches like crossbars [12], multiplexers [14] or decoupling networks [13] which themselves introduce more links into the system. Hai's scheme [19] has been shown to be superior to the other schemes. But it has the disadvantage that the spare node is a bottleneck. Also, the degree of the spare node is very high on account of the number of links incident to it.

The hierarchical networks discussed till now used clusters of size eight, i.e, they were based on a 3-cube or it's variant. The level-two network is a torus. Consider the diameter of such a scheme. The diameter of a hierarchical network based on a

torus of dimensions $M \times N$ as the level-two network is given by:

$$\begin{aligned} dia &= (torus\ diameter) \\ &\quad +(\#intermediate\ clusters\ on\ the\ longest\ path) \times \\ &\quad (\#links\ crossed\ in\ each\ cluster) \\ &\quad +\#links\ crossed\ in\ source\ cluster + \\ &\quad +\#links\ crossed\ in\ destination\ cluster \end{aligned}$$

The diameter of a torus D_t of width W is given by $2 \cdot \lfloor \frac{W}{2} \rfloor$. The torus will have $D_t - 1$ intermediate clusters on a path of length D_t. The above equation for the diameter then becomes

$$\begin{aligned} d &= 2 \cdot \lfloor \frac{W}{2} \rfloor \times (1 + Path\ length\ in\ each\ cluster) \\ &\quad + \#source\ cluster\ links + \#destination\ cluster\ links \end{aligned}$$

From this equation, it can be seen that the torus network contributes more towards the diameter. For a given network size (number of nodes), diameter can be reduced by reducing the dimensions of the torus. This can be achieved by increasing the cluster size. To see why, let C be the cluster size, and N be the total number of nodes in the architecture. Therefore, N/C represents the number of clusters required, and $W = \sqrt{N/C}$ is the width of the torus. The diameter of the torus is given by $2 \cdot \lfloor \frac{\sqrt{N/C}}{2} \rfloor$. As the cluster size C increases, the diameter decreases for the same network size. This increase in cluster size has to be balanced by several requirements, namely locality of communications, link cost, mean internodal

distance. etc. Also, by increasing the size of the basic cluster, we can also increase the number of spares in order to maintain the same spare to primary node ratio as before. This will also enhance the fault-tolerance.

In this chapter, we investigate the design of a hierarchical fault-tolerant interconnection network. Following the methodology of the previous chapter, we experiment with different basic blocks to come up with one which has good performance characteristics. We will use cost and LP ratio to evaluate different architectures.

This chapter is organized as follows. In Section 5.1, 5.2, and 5.3, we analyze the performance of hierarchical networks built using different clusters. Section 5.4 discusses the relative merits and demerits of the architectures, based on their performance evaluation. One scheme is selected based on it's performance and fault-tolerance. We investigate the properties of the proposed schemes in Section 5.5. The reconfiguration strategy in the presence of faults is discussed in Section 5.6. We discuss simulation results in Section 5.7. We compare the performance of the proposed architecture with that of the hypercube and the torus in Section 5.8. Section 5.9 concludes this chapter.

5.1 The Torus/4-cube(Perfect Embedding)

Banerjee and Peercy [16] proposed a fault-tolerant hypercube with spare CPUs attached to specific nodes. The nodes having spare CPUs (S nodes) were embedded in such a way that each primary node was a neighbor of atleast one S node, as shown in Figure 5.1. This was called a perfect embedding. We consider using a hypercube of dimension four for the cluster, in which four S nodes are attached using a perfect embedding. This cluster is denoted as the 4-cube(Perfect Embedding).

Table 5.1: Performance measures of the Torus/4-cube(Perfect Embedding) network.

M	N	Size	Links	Diameter	M.I.D	Cost	$L \cdot P$
2	2	80	112	9	4.80	36	536
2	4	160	224	11	6.05	44	1356
4	4	320	448	15	8.18	60	3668
4	8	640	896	19	10.33	76	9255
8	8	1280	1792	27	14.40	108	25807

These clusters are interconnected by a torus network at level-two, with the torus network replicated for fault-tolerance. The resultant hierarchical network is denoted as the Torus/4-cube(Perfect Embedding) network.

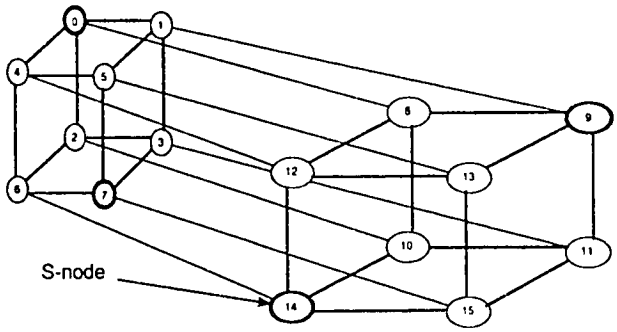

Figure 5.1: The 4-cube(Perfect Embedding), a hypercube based fault-tolerant basic block.

In Table 5.1 we give the performance measures of the Torus/4-cube(Perfect Embedding) network. Although a 4-cube has 16 nodes, we have counted the extra CPU in the S-nodes of the architecture as an extra node. Thus, a 4-cube(Perfect Embedding) is assumed to have 20 nodes instead of 16 nodes. This simplifying assumption is made so as to be able to compare architectures properly.

Table 5.2: Performance measures of the Torus/Augmented 4-cube network.

M	N	Size	Links	Diameter	M.I.D	Cost	$L \cdot P$
2	2	80	208	7	3.66	42	762
2	4	160	416	9	4.84	54	2016
4	4	320	832	12	6.02	72	5205
4	8	640	1664	16	8.54	96	14212
8	8	1280	3328	22	11.44	132	38086

5.2 The Torus/Augmented 4-cube Scheme

Hai [19] proposed a fault-tolerant architecture consisting of a hypercube of dimension d augmented by a spare. The spare is connected to all the nodes of the hypercube. This is called a Fault-Tolerant Basic Block of order d or a d-FTBB. Higher order cubes can be constructed by using the recursive construction property of the hypercube. For example, an m-cube can be constructed with d-FTBBs by joining together the corresponding nodes (excluding spares) of 2^{m-d} d-FTBBs. This is referred to as an *augmented* m-cube [19]. An Augmented 4-cube constructed using four 2-FTBBs is shown in Figure 5.2. In this figure, the corresponding nodes of four 2-FTBBs have been joined together to yield an Augmented 4-cube. The spare nodes in different FTBBs are not linked together.

We have explored the use of the Augmented 4-cube in a torus based hierarchical network. The hierarchical network built using the Augmented 4-cube is denoted as the Torus/Augmented 4-cube network. The performance of this architecture with respect to the important parameters is given in Table 5.2. When compared with the Torus/4-cube(Perfect Embedding), we can see that this architecture has a higher cost and LP product. This is because the 4-cube(Perfect Embedding) is topologi-

cally equivalent to a 4-cube. Therefore the Torus/4-cube(Perfect Embedding) is the same as the *HyperTorus* with a cluster size of 16 nodes. However, the Augmented 4-cube based architecture tolerates link failures, whereas the Torus/4-cube(Perfect Embedding) does not.

5.3 The Torus/4cube-4spare Scheme

In the previous scheme, the spares were part of each 2-subcube, and could only replace failed nodes in that subcube. This means that the spare utilization is not 100% because an idle spare in one subcube cannot replace failed nodes in another subcube. Banerjee [16] proposed a scheme which utilizes the perfect embedding to attach spare nodes to specific processors in a hypercube. We investigate a variant of the Banerjee's scheme, where the cluster size is sixteen nodes and each cluster has four spares [38]. The spares, instead of being attached to the primary nodes, are now part of a separate, fully connected 2-cube. This 2-cube is connected by four links to the main 4-cube. This is shown in Figure 5.3. This basic block is called the 4cube-4spare architecture. Separating the spares from the primary processors improves the fault-tolerance of the block because now there are extra links in the architecture which increases the number of spanning trees. Also, these spare nodes can function as interface nodes.

In Table 5.3 we present the performance results of the Torus/4cube-4spare network. As can be seen, this has lower diameter and mean internodal distance than the Augmented 4-cube based architecture. This is because the Torus/4cube-4spare has lesser number of links as compared to the Torus/Augmented 4-cube architecture. Due to this, the cost and LP product of the Torus/4cube-4spare is less

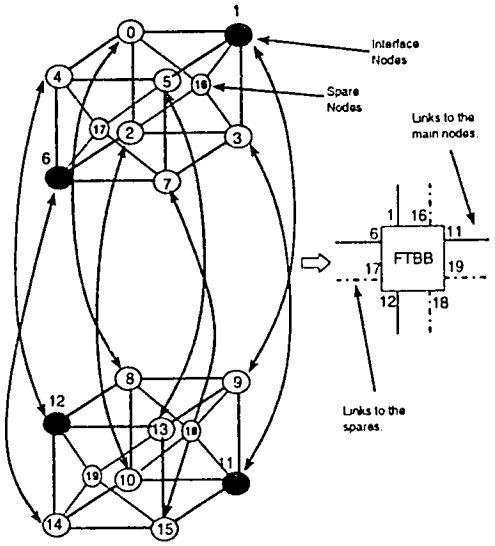

Figure 5.2: An Augmented 4-cube constructed using 2-FTBBs.

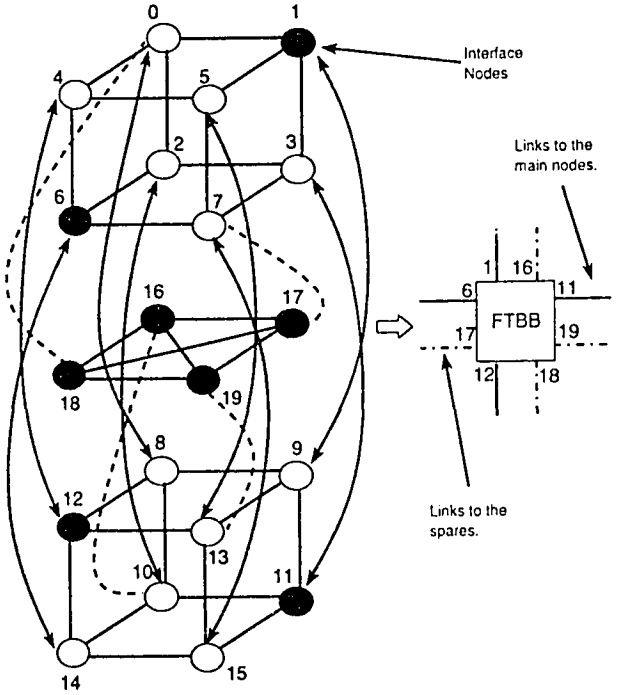

Figure 5.3: The 4cube-4spare, a 4-cube based FTBB having four spares.

Table 5.3: Performance measures of the Torus/4cube-4spare network.

M	N	Size	Links	Diameter	M.I.D	Cost	$L \cdot P$
2	2	80	184	8	3.80	40	703
2	4	160	368	10	5.20	50	1900
4	4	320	736	12	6.30	60	4621
4	8	640	1472	16	8.40	80	12390
8	8	1280	2944	20	10.40	100	30603

than the Torus/Augmented 4-cube scheme.

5.4 Results and Discussion

In this section, we compare and contrast the different architectures considered so far. The metrics used for comparison are Cost and *LP* ratio. In Figure 5.4, we show the relation between *LP* product and network size for the three architectures considered. As can be seen from the figure, the Torus/4-cube(Perfect Embedding) has the least *LP* product, followed by the Torus/4cube-4spare and the Torus/Augmented 4-cube. The Augmented 4-cube has more number of links than both the 4-cube(Perfect Embedding)and the 4cube-4spare. Therefore, the Torus/Augmented 4-cube has a higher *LP* product. As mentioned before, the Torus/4-cube(Perfect Embedding) is the same as the *HyperTorus*, with a cluster size of sixteen nodes instead of eight. Thus it has much better cost performance than both the Torus/4cube-4spare and the Torus/Augmented 4-cube. However, the 4-cube(Perfect Embedding) is expected to have a lower fault-tolerance and reliability than the 4cube-4spare scheme for the following reason. In the 4-cube(Perfect Embedding), the spare CPU in each S-node can only replace a failure which occurs in a neighboring node or the same node. This

means that a spare CPU can only replace failure which occur in d neighboring nodes, plus a failure of the main CPU itself. Whereas in the 4cube-4spare, a spare node can replace any failure, as long as the network is connected. Thus the Torus/4cube-4spare scheme is superior in this aspect to the Torus/4-cube(Perfect Embedding), inspite of having a higher LP product. The fault-tolerance and reliability of these schemes is calculated in Chapter 6.

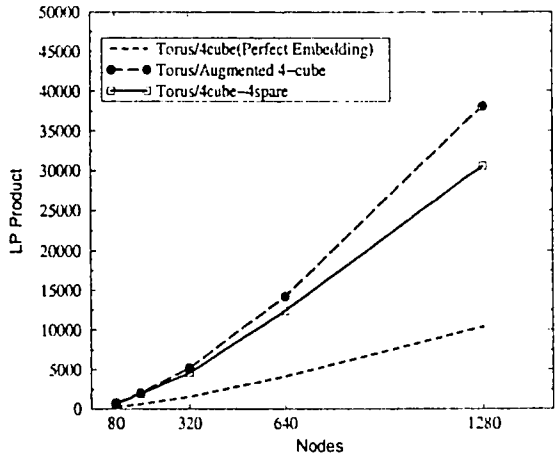

Figure 5.4: LP products of hierarchical fault-tolerant networks.

In Figure 5.5, we compare the costs of the hierarchical fault-tolerant architectures considered so far. Again, the Torus/4-cube(Perfect Embedding) has a lower cost than both the Torus/4cube-4spare and the Torus/Augmented 4-cube. However, the difference in cost between the Torus/4cube-4spare scheme and the Torus/4-cube(Perfect Embedding) is very small. Also, as mentioned above, the Torus/4cube-4spare has a better reliability than the Torus/4-cube(Perfect Embedding). In the next section, we give a formal construction methodology for the Torus/4cube-4spare

architecture and investigate it's properties.

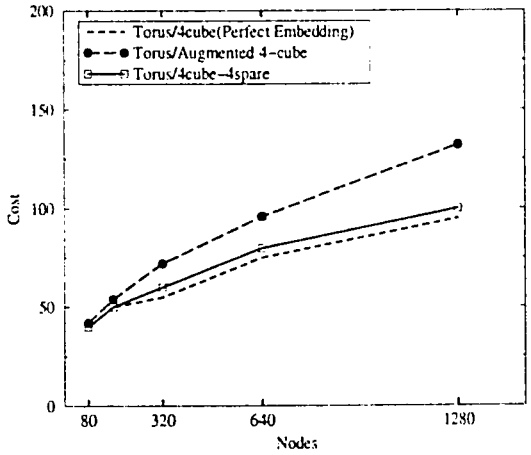

Figure 5.5: Costs of hierarchical fault-tolerant networks.

5.5 Properties of the Torus/4cube-4spare

The proposed scheme is a two-level hierarchical network. At the lowest level, we have a constant predefined fault-tolerant basic block (FTBB). At the second level, a 2D torus network is used to interconnect these FTBBs. If the size of the torus is $W \times W$, then we have W^2 FTBBs in the architecture. Such architecture is called a $W \times W$ FTBB architecture. Each node in the torus at level two is an FTBB. The FTBB used consists of a 4-cube, with a fully connected 2-cube of spare nodes for fault-tolerance. The spare nodes are connected to the main cube in such a way that every node in the main cube is at a distance of at most two hops from a spare, as shown in Figure 5.3. The nodes of the spare 2-cube and the four nodes of the main 4-cube which are attached to it (nodes 0,7,10,13) have an extra port.

The method of construction of higher order networks from the basic block is as follows. We use a 2D-torus at the second level to interconnect basic blocks. There are two networks, one for connecting the main nodes and the other for connecting the spares, as shown in Figure 5.6. Here, the solid lines are links connecting the main nodes, while the dashed lines are links connecting the spare nodes. Assuming that the torus is of size $M \times N$, we define the interconnection of the FTBBs as:

$$\begin{aligned}
Node\ 12\ of\ FTBB_{i,j} &\rightarrow Node\ 1\ of\ FTBB_{(i+1)\ mod\ M,j} \\
Node\ 18\ of\ FTBB_{i,j} &\rightarrow Node\ 16\ of\ FTBB_{(i+1)\ mod\ M,j} \\
Node\ 11\ of\ FTBB_{i,j} &\rightarrow Node\ 6\ of\ FTBB_{i,(j+1)\ mod\ N} \\
Node\ 19\ of\ FTBB_{i,j} &\rightarrow Node\ 17\ of\ FTBB_{i,(j+1)\ mod\ N}
\end{aligned}$$

In this architecture, there are eight interface nodes in each FTBB, four in the main cube and the four spares. Increasing the number of interface nodes as opposed to increasing the number of links, increases the fault-tolerant capability as well as reduces the inter-cluster and intra-cluster traffic density [35]. The interface nodes of the main cube participate in the torus at level-two. Also, the spares (which are also acting as interface nodes) participate in a torus at level two. Connecting the spare nodes in this way increases the fault-tolerance of the architecture. The reason is this: if those primary nodes which are also interface nodes fail, the FTBB will become disconnected from the network. Also, a spare node in a given FTBB can be used to replace a faulty node in any other FTBB (global sparing). However, it should be noted that this would complicate the reconfiguration strategy and the routing algorithm. Figure 5.7 shows an example of a 64-node network using FTBBs.

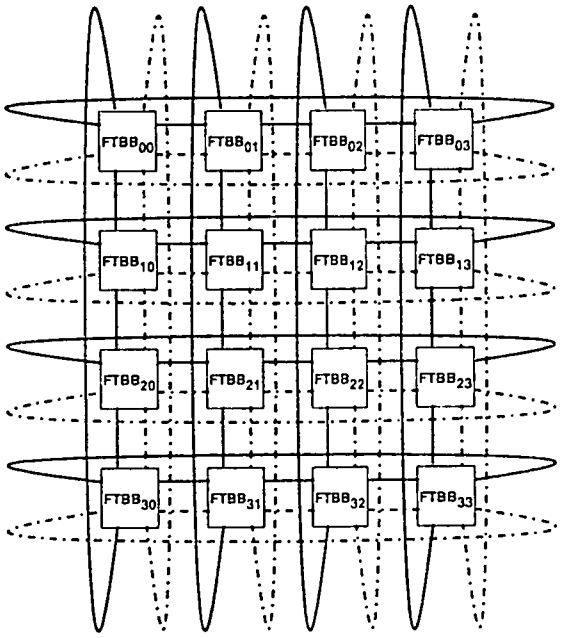

Figure 5.6: Interconnecting the FTBBs using a torus at level-two.

There are 16 nodes in each FTBB (excluding spares). Therefore, four FTBBs are needed for a 64-node architecture. The spares and the main nodes of the FTBBs are connected according to the rules given above. Thus the size of the level two network (i.e., torus) is 2×2. In this section, we discuss the properties of the proposed architecture. The FTBB of Figure 5.3 consists of a 4-cube, and a fully connected 2-cube as a spare block. There are eight interface nodes in the architecture, four in the 4-cube and the four spares. The interface nodes of the 4-cube are part of a torus network at level two (shown in solid lines). The spares are also a part of a separate torus network at level two (shown in broken lines).

Lemma 1 *The number of nodes in a* $W \times W$ *FTBB architecture is* $20 \times W^2$.

Proof: There are W^2 FTBBs in a $W \times W$ architecture. Each FTBB has 20 nodes,

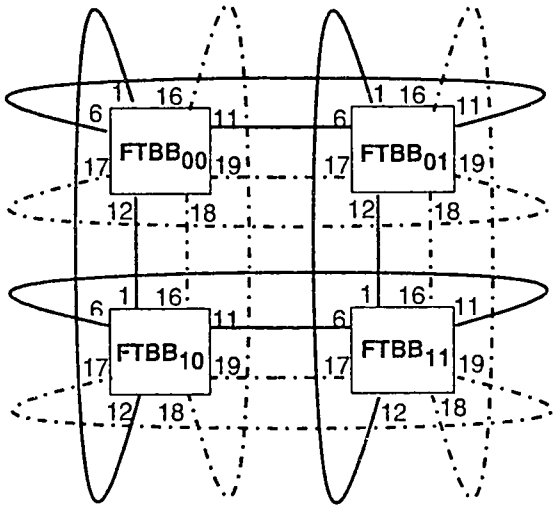

Figure 5.7: A 64 Node FTBB based hierarchical interconnection network.

16 main and 4 spare. Therefore, there is a total of $20 \times W^2$ nodes. ■

Lemma 2 *The number of links in a $W \times W$ FTBB architecture is* $42 \times W^2$

Proof: There are W^2 FTBBs in a $W \times W$ architecture. Each FTBB has 42 links - 32 links in the four cube, 6 links of the spare cube and 4 links connecting the 4-cube to the spare 2-cube. Therefore, there are $42 \times W^2$ links. ■

Theorem 2 *The diameter of a $W \times W$ FTBB architecture is* $8 \times \lfloor \frac{W}{2} \rfloor - 1$

Proof: The number of links on a path from a source to destination is given by the following components:

- The diameter of the torus $D_T = 2 \times \lfloor \frac{W}{2} \rfloor$.
- The number of links crossed in intermediate FTBBs $L_I = 3$.
- Number of intermediate FTBBs on a path from source to destination $N_{I-FTBB} = D_T - 1$.

- The number of links from an interface to a non-interface node in the source and destination FTBBs $L_S = L_D = 1$.

Combining all these components we get the diameter d as:

$$\begin{aligned} d &= (torus - diameter) + \\ &\quad (\#intermediate\ FTBBs) \times \\ &\quad (Path\ length\ in\ each\ FTBB)\ + \\ &\quad \#source\ FTBB\ links + \\ &\quad \#destination\ FTBB\ links \\ &= D_T + (D_T - 1) \times L_I + L_S + L_D \\ &= 2 \times \lfloor \frac{W}{2} \rfloor + 3 \times (2 \times \lfloor \frac{W}{2} \rfloor - 1) + 2 \times 1 \\ d &= 8 \times \lfloor \frac{W}{2} \rfloor - 1 \end{aligned}$$

■

Theorem 3 *The average path length in a $W \times W$ FTBB architecture is* $2 \times 2.09 + W/2$.

Proof: The average path length is composed of two components:

- Mean internodal distancein the FTBB = 2.09. This is obtained by using Equation 1.2
- The mean internodal distancein a $W \times W$ torus = $W/2$.

Each of the source and destination FTBBs contribute an average link distance of 2.09, whereas the torus links contribute $W/2$. ■

Table 5.4: Topological comparison of hierarchical fault-tolerant networks composed of the 4-cube(Perfect Embedding), the Augmented 4-cube, and the 4cube-4spare basic blocks.

	4-cube(Perfect Embedding)	Augmented 4-cube	4cube-4spare
Size	$20 \times W^2$	$20 \times W^2$	$20 \times W^2$
Degree	5	5	5
Diameter	$7 \times \lfloor \frac{W}{2} \rfloor + 3$	$10 \times \lfloor \frac{W}{2} \rfloor - 2$	$8 \times \lfloor \frac{W}{2} \rfloor - 1$
Cost	$5(7 \times \lfloor \frac{W}{2} \rfloor + 3)$	$5(10 \times \lfloor \frac{W}{2} \rfloor - 2)$	$5(8 \times \lfloor \frac{W}{2} \rfloor - 1)$

Similarly, we calculate the diameter and cost of the Torus/Augmented 4-cube and the Torus/4-cube(Perfect Embedding) networks. Table 5.4 compares the topological characteristics of the three proposed networks.

5.6 Reconfiguration Strategy

Consider the architecture shown in Figure 5.3. The failure of a single node can be tolerated in the following manner. A breadth-first spanning tree is built starting from a spare node [39, 40]. The expansion of a branch is stopped as soon as a neighbor of the failed node is added to the tree. As soon as all the neighbors of the failed node have been added to the tree, the process is stopped. Thus, the neighbors of the failed nodes may be the leaves or intermediate nodes of the spanning tree. A neighbor may also be a non-leaf node in the spanning tree if it lies on the path between another neighbor and the replacement. Now, these neighbors will route the information to the root of the spanning tree, which is the replacement for the failed node. The path the messages will take is the path to the root of the spanning tree.

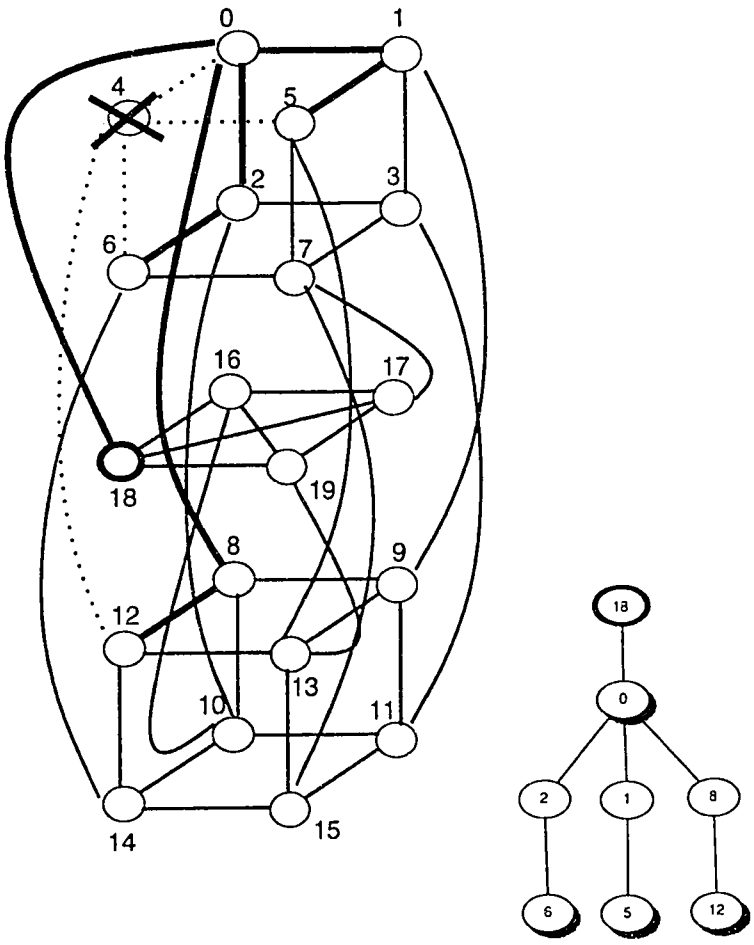

Figure 5.8: A fault and the resultant spanning tree used for reconfiguration.

An example is shown in Figure 5.8. Here, node-4 is assumed to have failed. The replacement chosen is node-18. It should be noted that node 18 is two hops away from node-4. We start building a spanning tree with node-18 as the root. The next node added is node-0, as it is one of the neighbors of node-4. We continue with the expansion and add other nodes. We find that the remaining neighbors of node-4 are reachable from node-0.

It should be noted that there might be as many spanning trees as there are

spare nodes. In this case, we choose the one which has the lowest mean internodal distance. This will minimize the message delay on the reconfigured path. In contrast to 3-FTBB scheme [19], more than one fault can be tolerated here, as there is more than one spare present. The reconfiguration strategy in case of multiple node failures will be similar to the single node failure case. We start building a spanning tree from a spare node, and stop as soon as the affected neighbors have been added. We may have to do this for each node which becomes faulty.

Now consider link failures. We start from one of the affected nodes, i.e. a node to which a failed link is incident, and build a spanning tree. We stop as soon as the node on the other side of the failed link is added to the tree. The new path to the neighbor will be the path along the spanning tree which leads to the neighbor. Figure 5.9 explains how this is done. The link joining node 0 and node 1 has failed. The reconfiguration process starts at node 0, and tries to build a spanning tree. This process stops as soon as node-1 is added to the spanning tree. After reconfiguration, the new path to node-1 is through nodes 18, 17, 7, 3 in that order. It might be that only primary nodes lie on this path. If this path goes through a spare node, then the spare will act only as a communication element, not as a computation element.

It should be noted that the reconfiguration strategy, in case of node and link failures introduces *dilation*, which means that the delay between two nodes increases because of the reconfiguration. In Figure 5.8 node-18 has replaced node-4. After reconfiguration, there is a path of length three to the neighbors of the failed node from the node which replaces the failed node.

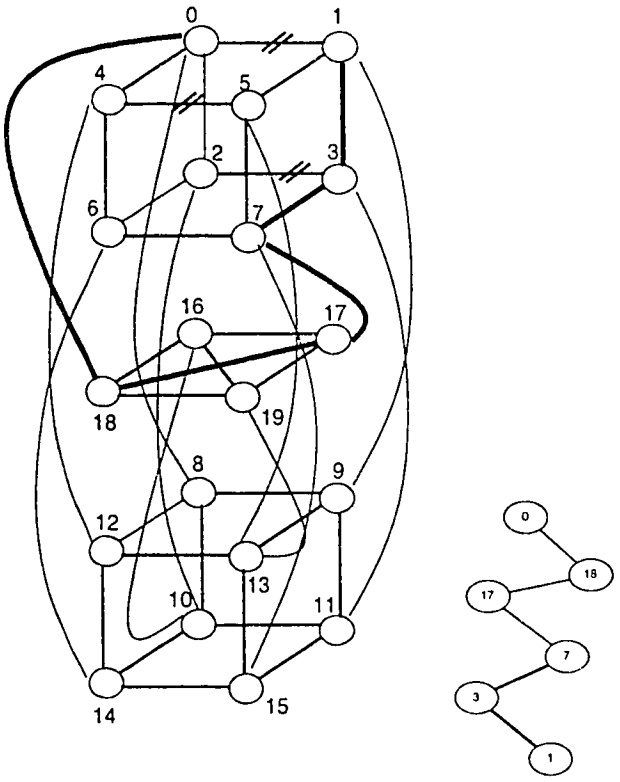

Figure 5.9: Reconfiguration under link failures.

5.7 Fault-Coverage Evaluation

In the following we show the results of simulations done to evaluate the reliability of the proposed FTBB for the node-failure model. We use fault coverage as the metric for simulation. We also compare the simulation results of the proposed FTBB with those of the hypercube and the torus.

5.7.1 Simulation Methodology

The simulation is done by exhaustively generating all the possible fault patterns (up to a certain number of faults) and calculating the number of cases where the network successfully reconfigures after each of these faults have occurred. For example, there are $^{20}C_4$ ways in which four nodes can fail in a FTBB of 20 nodes. Each one of these ways is called a fault pattern. Let $\mathcal{F}_k$ be the set of all fault patterns of length k. For example, $\mathcal{F}_1 = \{i | 0 \leq i \leq 19\}$ is a set of fault-patterns of length one. Each element of the set is the address of a failed node. Let $F_{i,k}$ denote the ith fault-pattern in $\mathcal{F}_k$. A fault is said to be recoverable if there is a path between every fault-free node in the network, otherwise the fault is unrecoverable. After every fault pattern $F_{i,k}$, $1 \leq i \leq (^{20}C_k)$, we check to see if all the non-faulty nodes in the network remain connected. After processing all $^{20}C_k$ fault patterns in $\mathcal{F}_k$, we calculate M_k, which gives the number of fault-patterns for which the network was not able to recover. Then, the percentage $(M_k) \times 100/(^{20}C_k)$ gives the fault-coverage of the architecture under k faults.

The fault coverage of the proposed basic block is compared with that of a 16-node hypercube and torus in Figure 5.10. As can be seen from this figure, for less number of faults, the fault-coverage of the hypercube, torus and the proposed

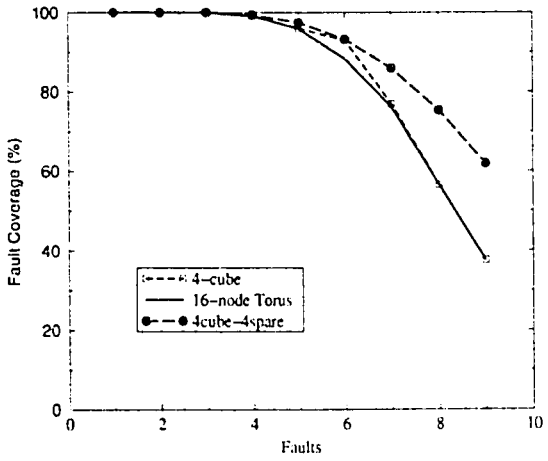

Figure 5.10: Fault-coverage comparison of the proposed 4cube-4spare with the hypercube and the torus.

FTBB are nearly the same. However, as the number of faults increase, the proposed FTBB recovers better from faults than the hypercube and the torus.

5.8 Cost and Performance Comparisons

In this section, we analyze the cost and performance of the proposed architecture. The proposed network is hierarchical in nature. It consists of hypercube based clusters connected using a torus topology at level-two. We would like to investigate how the proposed architecture compares to it's components, the hypercube and the torus. We use LP product and cost as a metric for comparison. For the purpose of this discussion, we have ignored the spare nodes in the FTBB. This is done so that the three architectures can be compared on an equal footing.

In Table 5.5 we show the LP product and the cost of the torus network for a

range of network sizes. A similar tabulation is done for the hypercube in Table 5.6. On comparing these two tables, we can see that the torus has a lesser cost than the hypercube for network sizes ranging from 64 to 256 nodes. Beyond those sizes, the high diameter of the torus offsets it's advantage of constant degree. Thus, for large sizes, the hypercube network has a lesser cost than the torus. However, the hypercube has more links than a torus of comparable size. Therefore, for network sizes ranging from 16 to 256 nodes, the torus outperforms the hypercube when LP product is used for comparison.

Table 5.7 gives the cost and performance of the proposed architecture. When compared with the corresponding tables for the torus and the hypercube, we can see that the Torus/4cube-4spare is more costly when LP product is used as the performance measure. Thus, there is a cost penalty associated with an increase in reliability.

The proposed architecture has cost comparable to that of the hypercube and greater than that of the torus for networks of sizes up to 128 nodes. As the network size increases, the torus has a cost disadvantage while the hypercube has a cost advantage. Since the proposed network uses hypercube based clusters, the cost of the proposed network is less than that of the torus and comparable to that of the hypercube for network sizes ranging from 128 to 1024 nodes. Thus, the proposed architecture is a viable alternative to the torus and the hypercube. It has a diameter comparable to that of the hypercube besides having the advantage of constant node degree.

Table 5.5: Cost and performance of the Torus.

Nodes	Links	dia	avg. dist.	LP	cost
64	128	8	4	512	32
128	256	12	6	1536	48
256	512	16	8	4096	64
512	1024	24	12	12288	96
1024	2048	32	16	32768	128

Table 5.6: Cost and performance of the Hypercube.

Nodes	Links	dia	avg. dist.	LP	cost
64	192	6	3.00	576	36
128	448	7	3.50	1568	49
256	1024	8	4.00	4096	64
512	2304	9	4.50	10368	81
1024	5120	10	5.00	25600	100

Table 5.7: Cost and performance of the Torus/4cube-4spare architecture.

Nodes	Links	dia	avg. dist.	LP	cost
64	184	8	3.82	703	40
128	368	10	5.16	1900	50
256	736	12	6.27	4621	60
512	1472	16	8.41	12390	80
1024	2944	20	10.39	30603	100

5.9 Concluding Remarks

In this chapter, we experimented with different fault-tolerant basic blocks and selected one on the basis of it's cost/performance parameters. The parameters used were LP product and cost. We discussed the properties of the proposed basic block in some depth. A reconfiguration strategy was proposed for the architecture. It's resilience in the presence of faults (fault-coverage) was evaluated and compared with that of the hypercube and the torus. It's cost and performance evaluation showed that it has higher cost than both the hypercube and the torus, which is a trade off for getting increased reliability.

In the next chapter, we shall develop analytical models to evaluate the reliability of the basic block. This will be then used to calculate the reliability of the entire network.

Chapter 6

Reliability Evaluation of the Proposed Architecture

In this chapter we present the network reliability and the task based reliability analysis of the basic block. We shall compare this with the reliability of the 16-node hypercube, Augmented 4-cube [19] and the 4-cube(Perfect Embedding) [16] respectively. Then, we shall calculate the reliability of the proposed hierarchical network.

The layout of this chapter is as follows. In Section 6.1 we calculate the task-based reliability of the proposed FTBB, the Augmented 4-cube and the 4-cube(Perfect Embedding)The results obtained here are used to calculate the reliability of hierarchical networks composed of these basic blocks in Section 6.2. The network reliability of the proposed scheme and that of the 16-node Hypercube are calculated in Section 6.3. This is followed by conclusions in Section 6.4.

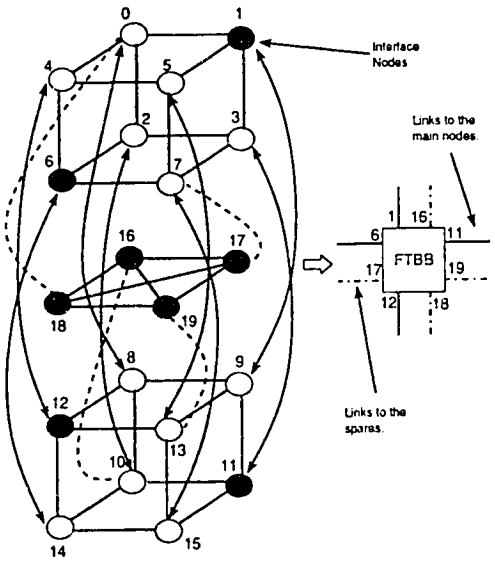

Figure 6.1: The 4cube-4spare Fault-tolerant basic building block (FTBB).

6.1 Reliability of the 4cube-4spare Basic Block

The basic block is a 16-node hypercube, connected to a 4-node completely connected graph which is acting as a spare block, as shown in Figure 6.1. It consists of 20 nodes, of which 16 are hypercube nodes and the remaining four are spare nodes. To calculate the reliability, we assume that in the architecture, any 16 nodes must function and can communicate with each other. This means that the network should remain connected. Using the combinatorial model we can write the reliability expression as:

$$\begin{aligned} R_{FTBB}(t) &= r_n(t)^{20} + {}^{20}C_1 r_n(t)^{19}(1 - r_n(t)) \\ &\quad + {}^{20}C_2 r_n(t)^{18}(1 - r_n(t))^2 \\ &\quad + {}^{20}C_3\, r_n(t)^{17}(1 - r_n(t))^3 \end{aligned}$$

$$+4818 r_n(t)^{16}(1 - r_n(t))^4 \tag{6.1}$$

were $r_n(t) = e^{-\lambda_n t}$ is the reliability of a node, and $\lambda_n = 10^{-5} hour^{-1}$ is the failure rate of a node.

The first term in the above equation stands for the event that all twenty nodes work. It has a coefficient of one ($^{20}C_0$). The second term is the probability of the event that one node has failed and nineteen are working. The coefficient of this term is twenty ($^{20}C_1$). The other terms are derived similarly. The coefficient of the last term is less than ($^{20}C_4$). The coefficient stands for the number of cases (of the occurrence of four faults) for which the network remains connected. This coefficient was calculated using the fault-coverage evaluation done in Section 5.7.

For a 16-node hypercube, since there are no spares, all the nodes should function in order that the cube should work.

$$R_{16HC}(t) \;=\; (r_n(t))^{16} \tag{6.2}$$

The reliability expressions for the other architectures are similarly derived. In the Augmented 4-cube, there are four 2-FTBBs. The probability of a 2-FTBB working is equal to the probability that any four out of the five nodes in the FTBB work. Therefore, the reliability of the Augmented 4-cube is equal to the probability that all the four 2-FTBBs work.

$$R_{AUG}(t) \;=\; (5\,(1 - R(t))\,R(t)^4 + R(t)^5)^4 \tag{6.3}$$

In the 4-cube(Perfect Embedding), each S-node can cover five faults, four in neigh-

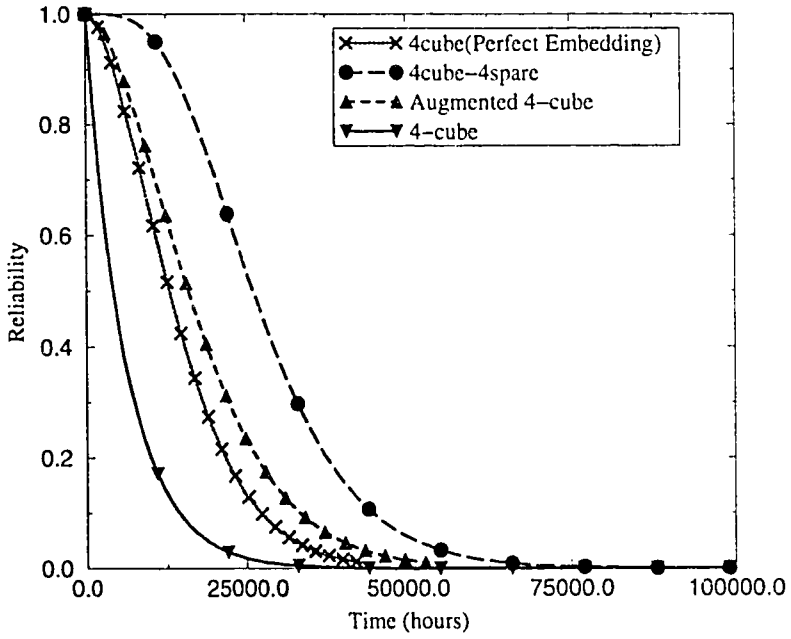

Figure 6.2: Task Based Reliability for the FTBB and the 16-Cube.

boring nodes and one in itself. This can be approximated as a parallel system, in which there are five nodes, and any one node should be non-faulty in order that the system should work. Since there are four such parallel systems in the 4-cube(Perfect Embedding), all of them should work if the system is to be non-faulty.

$$R_{PE}(t) = (6\,(1 - R(t))\,R(t)^5 + R(t)^6)^4 \tag{6.4}$$

A set of curves showing the reliability of the four configurations as a function of time is shown in Figure 6.2. If we integrate these expressions for $0 \leq t \leq \infty$ we get their MTTFs for the node failure model.

$$MTTF_{FTBB} = \int_0^\infty R_{FTBB}(t)dt$$

$$MTTF_{16HC} = \int_0^\infty R_{16HC}(t)dt$$

$$MTTF_{AUG} = \int_0^\infty R_{AUG}(t)dt$$

$$MTTF_{PE} = \int_0^\infty R_{PE}(t)dt$$

The node failure rate λ_n is assumed to be 10^{-5} $hour^{-1}$ [41]. Evaluating these expressions using $\lambda_n = 10^{-5}$ $hour^{-1}$, we get the following results:

$$MTTF_{FTBB} = 27916 \ hours.$$

$$MTTF_{16HC} = 6250 \ hours$$

$$MTTF_{AUG} = 18035 \ hours$$

$$MTTF_{PE} = 14714 \ hours$$

The MTTF of the perfect embedding is lower than the Augmented 4-cube, because it has one spare CPU covering five primary CPUs, whereas the Augmented 4-cube based scheme has one spare for every four nodes. The MTTF of the proposed scheme is the highest because there is full spare utilization within the FTBB.

6.2 Reliability of Hierarchical Networks

In the previous section, we calculated the reliability of a number of basic blocks, including the proposed 4cube-4spare basic block. MTTF values were obtained by integrating the reliability expressions. In this section, we use these MTTF values to calculate the reliability of hierarchical networks built using these blocks.

The level-two network is a Torus. To calculate the overall reliability of the hierarchical network, we calculate the failure rates of the basic blocks and use them to calculate the reliability of the level-two network. Note that each cluster at level-one is a node in the level-two network. The failure rate of that node is the failure rate of the cluster.

The effective failure rates of each of the basic blocks are obtained by taking the reciprocal of their MTTFs. The values are as follows:

$$\lambda_{FTBB} = 35.82 \times 10^{-6}/hour$$

$$\lambda_{16HC} = 160 \times 10^{-6}/hour$$

$$\lambda_{AUG} = 55.44 \times 10^{-6}/hour$$

$$\lambda_{PE} = 67.96 \times 10^{-6}/hour$$

These figures show that the proposed 4cube-4spare scheme and Augmented 4-cube have the lowest failure rate among the architectures considered. This is followed by the 4-cube(Perfect Embedding).

6.2.1 Subtask reliability for a Torus

Mesh and torii networks belong to a class of partitionable multicomputer networks. In such networks, a task only requires a certain number of components arranged in adjacent locations to function. The task requirements are specified in terms of submesh size. We use submesh reliability for the level-two torus network. Since a partition of a torus is a mesh, it is justified to use submesh reliability models here.

In [42], a submesh reliability model has been presented using the *consecutive n-out-of-N* system reliability model. This model is used to calculate the probability of atleast n consecutive components working out of a total of N aligned in a line. This model assumes that a submesh of the required size can always be recognized if it exists. The reliability of a *consecutive n-out-of-N* system is [42]:

$$R_{n|N} = 1 - \sum_{i=0}^{N} Z(i, N-i+1, n-1) p^i q^{N-i} \tag{6.5}$$

where

$$Z(\alpha, \beta, 1) = \begin{cases} \begin{pmatrix} \beta \\ \alpha \end{pmatrix} & for\ 0 \le \alpha \le \beta; \\ 0 & for\ \alpha > \beta. \end{cases} \tag{6.6}$$

$$Z(\alpha, \beta, \gamma) = \begin{pmatrix} \beta \\ i \end{pmatrix} Z(\alpha - \gamma i, \beta - i, \gamma - i) for\ \gamma \ge 2. \tag{6.7}$$

Here, p is the probability of a component working, $q(= 1 - p)$ the probability of component failure.

This equation is used to calculate the submesh reliability as follows. Consider a task requirement of $m \times n$ in a mesh of $M \times N$. We first consider a single row

of nodes, and calculate the exact probability of having m working nodes aligned consecutively in columns. Next, the reliability of n consecutive components, where each component represents a functional row is calculated. This is calculated as the probability of having atleast n components working. These working nodes may not be aligned to form a $m \times n$ submesh. The probability of having an alignment is extremely difficult to compute [43]. The complexity of this computation is approximately $O(N^2N^M)$ [42]. We therefore use approximations to solve the alignment problem. In a $M \times N$ system, a working $m \times n$ mesh can always be obtained irrespective of the alignment of working components if either of the following two conditions are satisfied:

1. There are atleast m consecutive rows, each having $\lceil \frac{n+N}{2} \rceil$ consecutive working components, or

2. There are atleast n consecutive columns, each having $\lceil \frac{m+M}{2} \rceil$ consecutive working components.

To obtain a tight bound, the minimum of the above two requirements is selected. If $Nm \leq Mn$ then a task requirement of $(m \times \lceil \frac{n+N}{2} \rceil)$ is used, otherwise a task requirement of $(n \times \lceil \frac{M+m}{2} \rceil)$ is used. An example illustrates this point. Assume a task requirement of a (60 × 70) submesh in a (100 × 80) mesh. Here, Mn=7000 and Nm= 4800. Since we have $Nm \leq Mn$, a task requirement of $(m \times \lceil \frac{n+N}{2} \rceil)$ is considered, which translates to a submesh size of (60 × 75).

6.2.2 Subtask Reliability Comparisons

We compare the reliability of different hierarchical networks using subtask reliability metric. The networks considered are the proposed network, the Perfect Embedding, the Augmented 4-cube, the 4cube based HIN, the 4-cube, and the BH/BH-FTBB-RS [19]. The BH/BH/FTBB-RS is a HIN, in which both the level-one and level-two networks are hypercubes. Each cluster in the HIN is a 3-FTBB. We consider a network of 1024 nodes and a task requirement of 512 nodes. For networks based on the torus at level-two, in which the cluster size is sixteen nodes, this translates into a requirement of a partition of dimension 4×8 out of a system of size 8×8. A similar approach is followed for the BH/BH-FTBB-RS HIN, in which the level-two network is a hypercube. The MTTF of the 3-FTBB has been calculated in [19]. From this MTTF value, we calculate the failure rate of the FTBB λ_{FTBB}. The subcube reliability $R_6scube(t)$ of a 6-subcube in a 7-cube has been calculated in [19]. We use this expression, with the failure rate equal to λ_{FTBB}, to calculate the reliability of a BH/BH-FTBB-RS network.

$$
\begin{aligned}
R_6scube(t) \;=\; & 128\,Rn(t)^{127} - Rn(t)^{128} + \\
& 7\left(64\,Rn(t)^{126} - 128\,Rn(t)^{127}\right) + \\
& 35\left(16\,Rn(t)^{120} - 96\,Rn(t)^{124} + 192\,Rn(t)^{126} - 128\,Rn(t)^{127}\right) + \\
& 21\left(4\,Rn(t)^{96} - 40\,Rn(t)^{112} + 160\,Rn(t)^{120}\right) + \\
& 21\left(-320\,Rn(t)^{124} + 320\,Rn(t)^{126} - 128\,Rn(t)^{127}\right) + \\
& 7\left(2\,Rn(t)^{64} - 24\,Rn(t)^{96} + 120\,Rn(t)^{112} - 320\,Rn(t)^{120}\right) + \\
& 7\left(+480\,Rn(t)^{124} - 384\,Rn(t)^{126} + 128\,Rn(t)^{127}\right) +
\end{aligned}
$$

$$35\left(8\,Rn(t)^{112} - 64\,Rn(t)^{120}\right) +$$
$$35\left(192\,Rn(t)^{124} - 256\,Rn(t)^{126} + 128\,Rn(t)^{127}\right) +$$
$$21\left(32\,Rn(t)^{124} - 128\,Rn(t)^{126} + 128\,Rn(t)^{127}\right) \tag{6.8}$$

6.2.3 Results and Discussion

In this section we compare the reliabilities of different hierarchical networks. In Figure 6.3 we plot the system reliability of the various hierarchical networks considered. As can be seen from Figure 6.3, the Torus/4cube network has the least reliability.

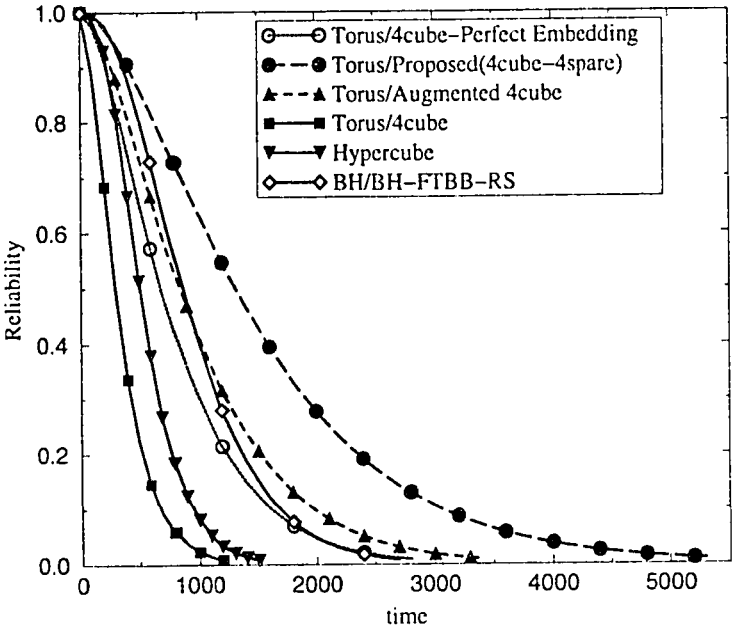

Figure 6.3: Reliabilities of different fault-tolerant hierarchical networks.

This is because it has no spare nodes for reconfiguration in case of node failures. The

Hypercube network has better reliability than the Torus/4cube scheme, because of the fact that it has more links. The proposed scheme and the Augmented 4-cube based network have a higher reliability than the Perfect Embedding. This is due to the fact that the basic block of the Perfect Embedding scheme has a lesser reliability than that of the basic blocks of the proposed scheme and the Augmented 4-cube scheme respectively. The Augmented 4-cube scheme has a lesser reliability than that of the proposed scheme because the spares in the proposed scheme can replace any failed node i.e., there is full spare utilization within the cluster.

6.3 Network Reliability

In the above model, we have put a condition that atleast 16 nodes should work and remain connected. This is basically Task-Based reliability. In this section we calculate the Network Reliability (NR) of the FTBB and 16-Hypercube. It is defined as the probability that every node in the network can communicate with every other node in the presence of link and/or node failures in the network.

The typical approach for the reliability analysis of a complex system is to decompose the system into smaller subsystems. Then, by the use of combinatorial analysis, fault-free analysis or a Markov modeling approach, the system probability of success is expressed in terms of probabilities of success of it's components. For a series-parallel structure, the analysis is simple. In a general network however, the probability of computing Terminal and Network Reliability is an NP-Complete [44, 45] problem. Methods for exactly computing NR in general networks require a list of spanning trees for the network. For a hypercube the number of spanning trees grows superexponentially with dimension n. However, it is possible to obtain

an upper or lower bound on a reliability measure more efficiently. The lower bound is of more interest as the system will be atleast this reliable. We do the lower bound network reliability analysis for the FTBB and the 16-HC.

Techniques for lower bounding the NR of hypercube networks have been studied by many researchers. Two techniques proposed in the literature [46, 47] require time exponential in the cube dimension n. We first review the better of these, a recursive technique by Bulka and Dugan [47] (henceforth called the BD approach). In the next section, we modify this technique to calculate the NR of the FTBB.

6.3.1 NR bounds for the Hypercube

Bulka and Dugan [47] presented a lower bound on the NR of an n-cube recursively as computed from a lower bound on the reliability of two component (n-1)-cubes. Let NR(n,p) denote the NR of an n-cube (C_n) with link reliability p. They used a C_2 as the basic cube for which the exact reliability is given as follows:

$$NR(2,p) = 4p^3(1-p) + p^4 \tag{6.9}$$

The reliability of a C_n is calculated from the reliability of a C_{n-1}. The C_n is decomposed into two congruent C_{n-1}s such that each node in one C_{n-1} is connected by an *exterior* link to it's congruent node in the other C_{n-1}. There are 2^{n-1} exterior links interconnecting the nodes in the C_{n-1}s. The links within each C_{n-1} are termed as *interior* links. The BD approach computes the reliability of a C_n by computing the probability of three disjoint cases of working and failed $(n-1)$-cubes and exterior links.

Case 1: Both $(n-1)$-cubes are operating and i exterior links operate, $1 \leq i \leq$

$2^{n-1} - 2$.

$$T1 = NR(n-1,p)^2 \sum_{i=1}^{2^{n-1}-2} \binom{2^{n-1}}{i} p^i \, q^{2^{n-1}-i}$$

Case 2: Atleast one (n-1)-cube is operating, and one exterior link has failed. The node at the endpoint of the failed link in the non-operating C_{n-1} is linked to atleast one of it's neighbors by an interior link.

$$T2 = [2.NR(n-1,p)(1-q^{n-1}) - NR(n-1,p^2)] \times 2^{n-1} p^{2^{n-1}-1} q$$

Case 3: All 2^{n-1} exterior links operate . Let p' = $p^2 + 2pq$.

$$T3 = NR(n-1,p')p^{2^{n-1}}$$

The lower bound on the NR of the n-cube is then:

$$NR(n,p) = T1 + T2 + T3 \tag{6.10}$$

6.3.2 NR Bounds for the 4cube-4spare

In this section, we will develop lower and upper bounds on the network reliability for the proposed building block. The lower bound is developed as follows. We count the number of spanning trees in the architecture. This is done by using decomposition. The number of spanning trees in the architecture, T_{ftbb} is the product of:

- Number of spanning trees in the 4-cube - T_4.

- Number of spanning trees in the fully connected subnetwork of spare nodes, i.e. 12.
- The number of links connecting the 4-cube to the spare nodes, i.e. 4 links.

The number of spanning trees T_d in a d-dimensional hypercube is given by:

$$T_d = 2^{-n} \prod_{j=1}^{n} (2j)^{\binom{n}{j}} \tag{6.11}$$

The number of spanning trees in a 4-cube is

$$T_4 = 2^{-4} \prod_{j=1}^{4} (2j)^{\binom{4}{j}} \tag{6.12}$$

Therefore, the number of spanning trees T_{ftbb} is given by:

$$T_{ftbb} = T_4 * 4 * 12 \tag{6.13}$$

For the network to remain connected, atleast one of these spanning trees should work. Thus, we use the combinatorial model to get the probability of one spanning tree working, out of a total of T_{ftbb} trees. This gives the lower bound on the reliability of the network $NR_{lower}(t)$.

$$NR_{lower}(t) = 1 - (1 - TR(t))^{T_{ftbb}} \tag{6.14}$$

Here, $TR(t)$ is the reliability of a tree. It is the probability that all the links in a tree are working. Since there are 19 links in a spanning tree having 20 nodes, we get the reliability of a spanning tree as:

$$TR(t) = R_l(t)^{19} \tag{6.15}$$

where $R_l(t) = e^{-\lambda_l t}$ is the reliability of a link. The failure-rate of a link, $\lambda_l = 10^{-6}$ / *hour* [41].

For the upper bound, we calculate the minimum number of links that should work so that the network remains connected. The probability of these links working gives the upper bound on the reliability. A four dimensional hypercube has a link connectivity (i.e., the minimum number of links that should fail in order to partition the network) of four. This is also true for the proposed FTBB. This can be verified by noting that failure of the four links connecting the 4-cube to the four spares will disconnect the spare nodes from the main nodes. The reliability of the network is then

$$NR_{upper}(t) = \sum_{i=17}^{42} \binom{42}{i} R_l(t)^i \left(1 - R_l(t)\right)^{42-i} \tag{6.16}$$

In Figure 6.4 we plot the network reliability bounds for the FTBB and the network reliability of the 4-cube. We can see that the proposed FTBB has better reliability than the 4-cube. This is because the proposed network has more spanning trees, and therefore has a higher probability of remaining connected in the event of failures.

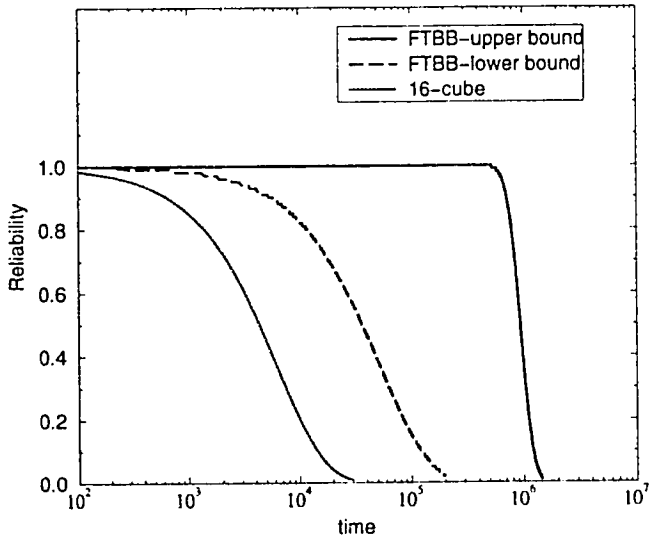

Figure 6.4: Network reliability bounds for the hypercube and the 4cube-4spare

6.4 Concluding Remarks

In this chapter, analytical models were developed to evaluate the reliability of the proposed hierarchical network. First the reliability of the basic block was calculated. This was used to calculate the reliability of the overall network. Comparisons were made with other architectures. A Network Reliability model for the basic block was also developed. It was shown that the probability of the network remaining connected in the case of faults is higher in the proposed 4cube-4spare basic block, as compared to the 4-cube.

Chapter 7

Routing Algorithms for the Proposed Architectures

Some of the proposals for fault-tolerant hypercubes made the assumption that a failed node can only be replaced by an adjacent spare node [13, 14, 15]. It was also assumed that the spare which replaces a primary node has to be adjacent to the neighbors of the primary node. In other words, the physical hypercube topology was sought to be preserved in all cases of failures. Due to this, these architectures had a lot of extra components, which attempted to reconfigure the architecture in case of failures and preserve the topology, resulting in a very high cost.

In this work, we transfer the responsibility of preserving the topology to the routing algorithm. Thus, we attempt to preserve the logical topology, and not the physical topology.

In this chapter, we give an outline of the routing algorithm for the proposed FTBB. The algorithm is fault-tolerant, in that it attempts to find a path to the destination if one exists. We also propose a routing algorithm for the *HyperTorus*

architecture discussed in Chapter 4.

The layout of this chapter is as follows. Section 7.1 gives an overview of the e-cube routing algorithm. Section 7.2 describes the fault-tolerant e-cube routing algorithm. A routing algorithm for the proposed basic block is described in Section 7.3. Section 7.4 gives a routing algorithm for the *HyperTorus* architecture. Section 7.5 concludes this chapter.

7.1 The e-cube Algorithm

The *e*-cube or "left-right" (LR) algorithm is the simplest of all hypercube routing algorithms. This algorithm works as follows: assume that there is a message from source-S destined for destination-D. The n network dimensions are strictly ordered as d_{n-1}, d_{n-2}, ..., d_0 using the relation $d_i > d_j$ if and only if $i > j$. The dimensions are inspected from highest to lowest in order, and links in dimension d_i are traversed if and only if the destination address differs from the current node address in that dimension [48]. While this algorithm allows only one path for a given source-destination pair, it ensures deadlock-free routing for binary hypercubes. This is because it enforces an order in which a message can access links.

7.2 Fault-Tolerant Routing Algorithm for Binary Hypercubes

Youran Lan [49] proposed a software based oblivious routing algorithm for hypercubes. It is a fault-tolerant algorithm which attempts to find a route to the desti-

nation in the presence of faults. The path chosen in the presence of faults may be non-optimal.

This algorithm works as follows. In the absence of faults, the algorithm is equivalent to the e-cube routing algorithm for hypercubes. It uses dimension ordering to find a path to the destination. In the presence of faults, however, a non-minimal path is chosen. This non-minimal path attempts to navigate around the faulty region. The detailed steps are given in Figure 7.1. The header of the message consists of three parts. The first component, u_d is the destination address. A dimension is said to be faulty if either the neighboring node at that dimension is faulty or the link at that dimension is faulty. A *preferred dimension* is one which is used to route a message to the destination when minimal routing is used. At any node, all the faulty preferred dimensions are called blocking dimensions. The second component of the header T, is a routing tag, which keeps track of the blocking dimensions encountered. It is reset to zero initially. The third part is the message data. This algorithm works if the number of faulty components (node and/or links) is less than the dimension of the hypercube. This condition ensures that the hypercube remains connected in the presence of faults. Otherwise, in a n-cube, failure of all the links connecting a node will partition the cube into two disjoint subnetworks and the algorithm will fail.

Example 7.1: In Figure 7.2, we show how routing takes place in the case of failure of intermediate links and/or nodes. Here, the destination node is assumed to be non-faulty. Also, the network is assumed to be connected. The hypercube is of dimension four and the source and destination nodes are 0 and 15. Initially, the path taken is the same as that under dimension order routing. At node-3, no more progress can be

algorithm : *HC-Route*
Input :
Local address (u_0)
Destination address (u_d)
Routing tag (T)
Incoming dimension (w)
Output :
A selected dimension to forward a message
begin
$r = u_d \oplus u_0$;/*Calculate relative address*/
FOUND = FALSE;
if($u_d = 0$) then *destination reached?*
{ send message to local processor; stop;}
try and find a non-faulty outgoing dimension on which to route the message
for(i=0; $i < n$; i++)
if(($r_i = 1$) AND (dimension i is non-faulty) AND ($i \neq w$))
{ l=i; FOUND = TRUE; break; }
check if the incoming dimension is the preferred dimension
if ((FOUND=FALSE) AND ($r_w = 1$)) then
{ $l = w$; FOUND=TRUE; break; }
Try and find a spare dimension
If (FOUND=FALSE)
record blocking dimension in T
for(i=0;$i < n$;i++)
if($r_i = 1$) $t_i = 1$
find an unused non-faulty spare dimension
for(i=0;$i < n$;i++)
if(($t_i = 0$) AND (i is non-faulty)) then
$l = i$; $t_l = 1$;/*update routing tag*/
send the message (u_d,T,data) to the lth dimension;
end

Figure 7.1: The oblivious fault-tolerant hypercube routing algorithm.

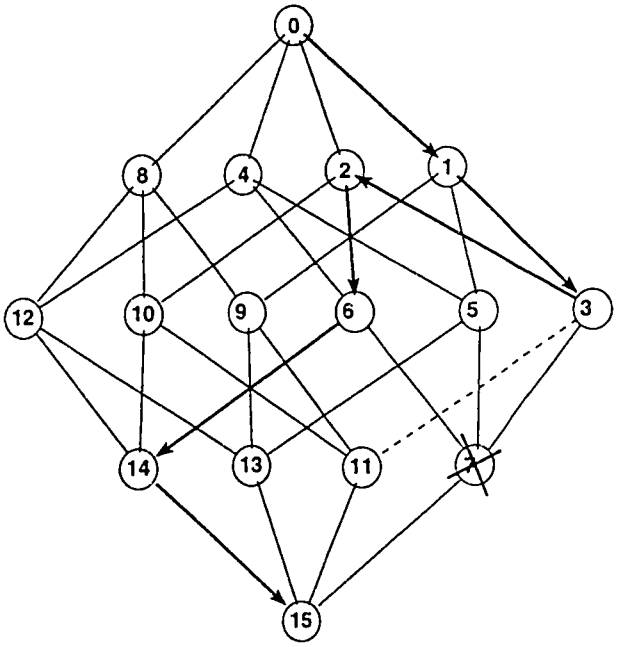

Figure 7.2: Routing in a hypercube in the presence of faults.

made as the two outgoing dimensions are faulty. Therefore, an unused dimension, which leads to node-2 is used. From node-2, the routing is normal dimension order. The dimensions used are respectively $(3, 4, 1)$ in that order.

7.3 The 4cube-4spare Routing Algorithm

Routing in the FTBB is more complicated than it is for an n-cube. This is because the following conditions have to be taken care of:

- A failed primary node is replaced by a spare node. Thus, any message destined for the failed primary node must now be directed to the spare node which replaces it.

- Consider a message destined for a non-faulty node N. Assume that the message has now arrived at an intermediate node I which was previously connected to N but is now disconnected due to node/link failures. However, this node N is still connected to the graph through the fully connected 2-cube network of spare nodes. Thus, node I must direct the message to a spare node, which will route the message properly to node N. This is shown in Figure 7.3. Here the subcube consisting of nodes 0 and 2 is isolated because of link failures. If a message destined for node 2 is received by node 5, it should be forwarded to the spare node 18 which can forward it to node 2.

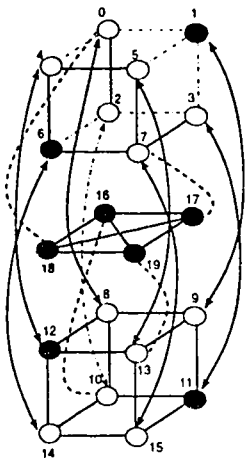

Figure 7.3: Routing in case of network partitioning.

The message header is modified for FTBB routing. Besides the fields mentioned before, the message header also contains the following fields:

Visited: a set which contains the addresses of spare nodes which have been already visited.

Visiting: a set which contains the address of the spare node which is currently being visited.

VisitedCount: an integer which contains the count of the spare nodes visited so far. It is equal to the cardinality of *Visited.*

The routing algorithm for the FTBB operates in two phases. In the first phase the algorithm is similar to the algorithm shown in Figure 7.1. In this phase, a path to the destination node is attempted. This phase will succeed if a path to the destination exists in the 4-cube. However, if node/link failures have caused the FTBB to be partitioned so that the destination is in a separate partition which is connected to the 4-cube via spare nodes, the algorithm will fail. Then the algorithm goes into the second phase. In this phase, the algorithm sends the message to a spare node. It is assumed that the spare node which replaces a failed primary node inherits it's address. When a message arrives at a spare node, the header is examined. If the destination address is same as the address of the failed node which has been replaced by the current spare node, it is absorbed. Otherwise, if the destination node is connected to the spare, then it is routed appropriately. In case both these steps fail, the message is sent to the next spare node. Thus, the primary nodes and spare nodes run different routing algorithms.

To distinguish between the two phases of the algorithm, we use a *Type* flag. This flag is added to the header of the message to enable the intermediate nodes on a path to distinguish which phase a message is going through. If it is searching for a primary node and has not encountered a dead-end yet, then the *Type* flag has a value 'P', otherwise it hs a value 'S'. The steps taken are given in Figure 7.4

```
algorithm : FTBB-Route
Input :
    Local address (u_0)
    Destination address (u_d), Routing tag ( T), Flag ( Type)
    Incoming dimension (w)
Output :
    A selected dimension on which to forward a message
begin
    Phase One
    if(Type = 'P') then
        r = u_d ⊕ u_0:/* relative address calculation */
        if(u_d = 0) then /* Destination reached */
            { send message to local processor; stop; }
        try and find a non-faulty, outgoing dimension 'l'
            on which to route the message
        If an outgoing dimension cannot be found, then
        check if the incoming dimension 'w' can be used.
        if the incoming dimension 'w' can be used, then let l = w.

        If the incoming dimension cannot be used, then
        Try and find a unused spare dimension s
        If such a dimension can be found, then mark s as used in T and set l=s
        If(a dimension l has been found)
            send the message (u_d,T,data) to the lth dimension;
        else
            Go into Phase II
            Type = 'S';
            T = 0; /* mark dimensions unused */
    Phase Two
    if(Type = 'S') then
        If (a spare node is directly connected) then
            send message to the spare node
        else
            find a non-faulty outgoing dimension (non-incoming) and
            route the message along it to the neighboring node
end
```

Figure 7.4: The oblivious fault-tolerant FTBB routing algorithm.

The spare nodes execute a different algorithm. This is because a message will come to the spare nodes only if a path to the destination was not found in the 4-cube. To avoid livelock, we maintain a list of spare nodes visited. A spare node is said to be visited, if it is the destination of the message or if the subnetwork connected to the spare node is visited. Routing is said to fail, if all the spare nodes are visited and the destination is not reached yet. Once the message reaches the subnetwork connected to the spare node, it again follows the dimension order routing algorithm. Figure 7.5 shows how this is done. Assume for a moment that

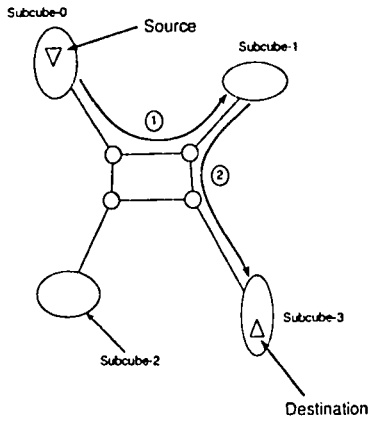

Figure 7.5: Routing in the FTBB in the presence of faults.

failures have partitioned the original FTBB, so that there are now four subcubes of order two interconnected through spare nodes. This can occur due to link failures. A message is sent by a source in Subcube-0 to a destination node in Subcube-3. The links between the subcubes have failed, so the only connection between them is through the spare nodes. The nodes in Subcube-0 first try to find a path to the destination. Since no such path exists, the message is sent to the nearest spare node (Phase-II).

A spare node might have either replaced the destination node, or might be connected to the destination node. If the spare node has replaced the destination node, the message is absorbed. Otherwise, the subnetwork connected to the spare node is visited. If all the spare nodes are visited without the destination being reached, then the algorithm signals failure.

Example 7.2: When the source and destination nodes are nonfaulty and all the primary nodes in the network are connected to each other, the routing is the same as the normal fault-tolerant routing which is discussed in Figure 7.1. Figure 7.2 shows how the algorithm works. In this case, the spare nodes are not visited at all.

Example 7.3: When a primary node fails, it is replaced by the spare node. Assume node-7 has failed and has been replaced by spare node-17. Figure 7.7 shows how routing occurs when node-0 wants to send a message to node-7. The algorithm first tries to find a path to the destination using fault-tolerant routing. In this case, no path exists, because the destination node has failed. So, the algorithm fails when the message reaches node-15 after trying all possible steps. At this point, the algorithm goes into phase-2, where it searches for the spare node which replaced the failed node. The *Type* flag is initialized to 'S' to indicate that the algorithm is now in phase-2. The message is now sent to a primary node which is connected to a spare node. Here, it is routed to the spare node. In the algorithm, we can see this happening when node-15 forwards the message to node-14, then to node-10, which finally delivers the message to node-16. Incidentally, this is the replacement of the failed node. If it were not the replacement of the failed node, the message would have been forwarded to the next spare node.

```
algorithm : FTBB-Route-Spare
Input :
    Local address (u_0)
    Destination address (u_d), Routing tag ( T), Flag ( Type)
    Sets Visited,Visiting
    int Visited-count
    Incoming dimension (w)
Output :
    A selected dimension on which to forward a message
begin
    if(VisitedCount = 0) /* going to the spare nodes for the first time?  */
        Visited ← Visited ∪ {u_0}
        ++VisitedCount;
    Has the current node replaced u_d?
    if (the current spare has replaced u_d)
        absorb the message
        stop
    /* returning from an unsuccessful search into a subnetwork
    associated with the current node
    if(u_0 ∈ Visiting)
        Visited ← Visited ∪ u_0; /* mark u_d visited */
        ++VisitedCount;
        have all the spare nodes been visited?
        if(VisitedCount = 4)
            Signal Routing Failure
        else
            send message to next spare node
            stop
    Explore Subnetwork connected to the spare
    Type ←' P' /* In the subnetwork, follow the fault-tolerant algorithm*/
    T ← 0 /* initialize routing tag */
    Visiting ← Visiting ∪ u_0;
    Send message to a connected primary node.
end
```

Figure 7.6: The routing algorithm executed by a spare node in a FTBB.

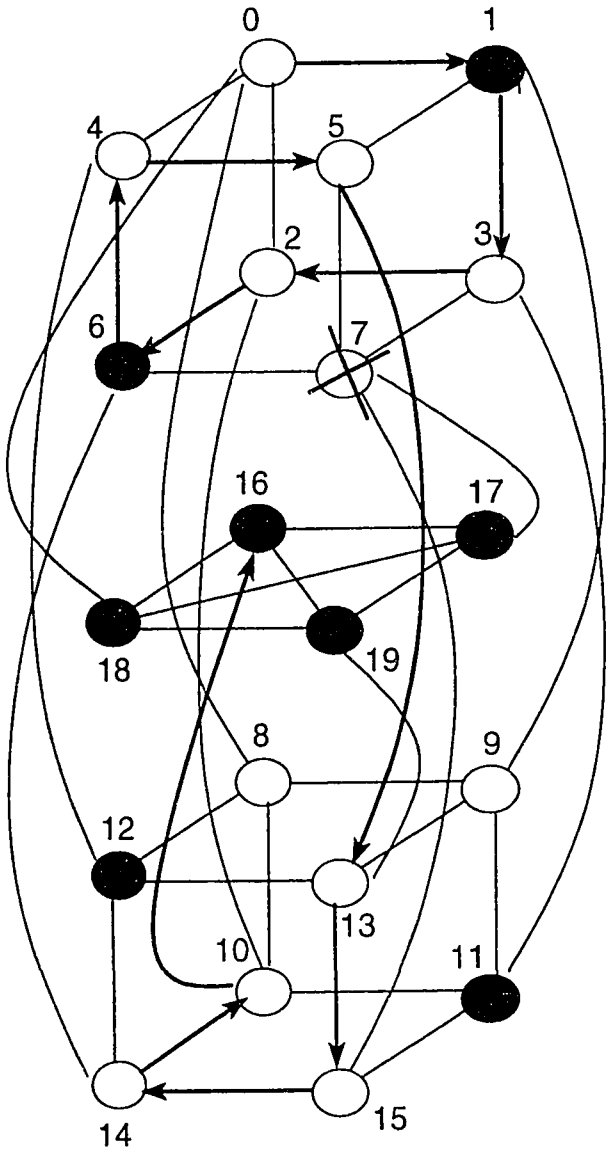

Figure 7.7: Routing when a spare node has replaced a primary node.

The complexity of the algorithm can be analyzed as follows. First, note that every time a message takes a spare dimension, it's distance to the destination increases by two. This is because a spare dimension is used for backtracking if all the preferred dimensions with respect to the current node are blocked. Thus, the spare dimension increases the path-length by one. Also, when a spare dimension is taken, some other dimension will become preferred. Thus, the number of preferred dimensions increases by one. This means that the remaining path to the destination has increased by one. This, coupled with the fact that backtracking along the spare dimension increases the path length by one, means that the use of a spare dimension increases the path length by two.

Note that fault-tolerant hypercube routing algorithm requires that there should not be more than $(n-1)$ faults in an n-dimensional hypercube. This means that there can be up to $(n-1)$ faults. Therefore, in the worst case, the algorithm will take $H(s,d) + 2(n-1)$ steps to route the message from source s to destination d, where $H(s,d)$ is the hamming distance between s and d. If the number of faults increases, then the routing algorithm will enter Phase II, and the number of steps needed will depend on the exact pattern of faults.

7.4 Routing in the *HyperTorus*

The *HyperTorus* was discussed in Section 4.3. This architecture consisted of hypercube based clusters connected using a torus at level-two. In this section, we propose a routing algorithm for this architecture. The routing algorithm for this architecture is an adaptation of the standard torus routing algorithm, which is deadlock-free [50]. The modified algorithm accounts for the fact that there are two torus networks

and the nodes in these are hypercube clusters. We assume that the address of any node in the architecture is given by $< X, Y, N >$, where (X, Y) is the address of the cluster, and N is the address of the node within the cluster.

In dimension ordered routing, we first route the message along the X direction until the correct column is reached. Then the message is routed in the Y direction to the destination node. Since there are two torus networks, we use one network for routing in the negative X and Y directions and the other for routing in the positive X and Y directions. Figure 7.8 shows the detailed routing in *HyperTorus*.

Example 7.4: We show an example of how this occurs in Figure 7.9. The dimensions of the torus are 4 × 4. The source node is $< 0,0,3 >$ and the destination node is $< 2,2,7 >$. According to the algorithm, messages in the positive X and Y directions are routed in the network denoted by solid lines in the figure. How the interface nodes are chosen depends on the offset. The offsets are tested in the order (X^-, Y^+, X^+, Y^-). Since the destination is in the positive Y direction (Y^+), the interface node associated with this is chosen, which is node-2. This continues, until the Y-offset becomes zero. At this point (cluster (0,2)), there is a change in direction to X^+. Interface node-1 is chosen to forward the message to the next cluster. In the destination cluster, the message is forwarded to the appropriate node using the hypercube routing algorithm.

The Torus/4cube-4spare network is very similar to the *HyperTorus*, except for the fact that we now have FTBBs instead of hypercubes as clusters and not all nodes are interface nodes. Thus, routing for this architecture will be a combination of Torus and the FTBB routing algorithms. When a message reaches an intermediate FTBB, the torus routing algorithm will be used to find the next interface node to

algorithm : HyperTorus *Route*
Input :
Local address (X_c, Y_c, N_c)
From Message Header
Destination address (X_d, Y_d, N_d)
Relative address of Interface-node R
Output :
A selected dimension l on which to forward a message
begin
$\delta_x = X_d - X_c$; $\delta_y = Y_d - Y_c$
if ((| δ_x |> 0) OR (| δ_y |> 0)) /* *intermediate cluster* */
if ($\delta_x < 0$) $R \leftarrow N_c \oplus 7$
else
if ($\delta_y > 0$) $R \leftarrow N_c \oplus 2$
else
if ($\delta_x > 0$) $R \leftarrow N_c \oplus 1$
else
if ($\delta_y < 0$) $R \leftarrow N_c \oplus 4$
if ($R = 0$) /* *desired interface node reached* */
$l \leftarrow 4$
else /* *destination cluster* */
$R \leftarrow N_c \oplus N_d$
if $R = 0$ /* *destination reached* */
Absorb message
Stop
/* *use hypercube routing to route the message to the destination* */
for ($i = 0; i < 3; i++$)
if($R_i = 1$) { $l \leftarrow i$; break; }
Route the message along dimension-l.
end

Figure 7.8: The *HyperTorus* routing algorithm.

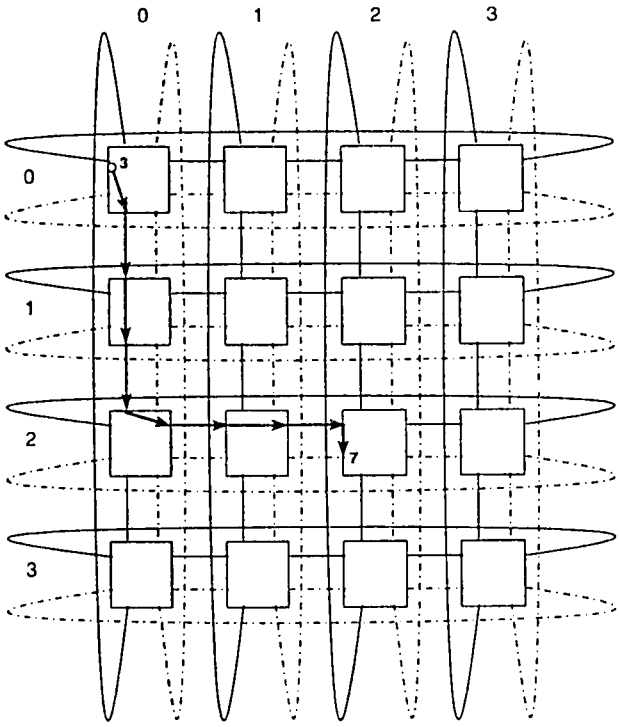

Figure 7.9: Routing in the *HyperTorus*.

send the message to. The FTBB routing algorithm will be used to compute a path to the interface node.

7.5 Concluding Remarks

In this chapter, we proposed a fault-tolerant routing algorithm for the basic block. The proposed algorithm was shown to find a path to the destination in the presence of faults, if one exists. A routing algorithm for the *HyperTorus* has also been developed. In the next chapter, we conclude this thesis and give directions for future work.

Chapter 8

Conclusions and Future Work

In this work, a new hierarchical interconnection network was designed. Various alternatives were examined and evaluated, based on their cost and performance. A hierarchical interconnection network called the *HyperTorus* was proposed. It is a two-level design, in which the level-one network is a hypercube, and the level-two network is a torus. In this design, an attempt has been made to strike a balance between cost and performance, among a number of architectural alternatives considered.

The same procedure was undertaken to propose the Torus/4cube-4spare, a hierarchical fault-tolerant interconnection network. It is a two-level network, in which the level-one network, the 4cube-4spare is a hypercube based fault-tolerant block, and the level-two network is a torus. The properties of the proposed architecture have been studied. The resilience of the basic block in the presence of faults has been analyzed and compared with that of the hypercube and the torus. It has been found that the fault-tolerance of the proposed 4cube-4spare basic block is better than that of the hypercube and the torus. A performance comparison was also made using *LP*

product as the performance measure. It was found that the proposed architecture has a higher LP product than both the hypercube and the torus. This is the penalty paid for increased fault-tolerance. In addition, it has been shown that the cost of the proposed architecture is comparable to that of the hypercube and better than the cost of the torus for a range of network sizes from 128 to 1024 nodes.

An algorithm has been developed, which allows the architecture to be reconfigured in the presence of faults. This algorithm finds a path to the destination if one exists. Also, when a spare replaces a failed primary node which was the destination of a message, this routing algorithm finds the destination successfully.

The reliability of this architecture has been calculated and compared with that of some existing architectures. It has been shown that the proposed architecture has better reliability than a number of contemporary architectures. Network reliability calculations were done for the proposed architecture.

8.1 Conclusions

In this work, we have used the technique of hierarchical construction to come up with networks having low link cost and good performance. We have proposed a hierarchical interconnection network, which incorporates the desirable properties of the hypercube and the torus. It has a fixed degree like the torus networks, and a cost comparable to that of the hypercube. The proposed network minimizes link cost, and gives better performance as compared to the Torus/Möbius-cube and the Torus/Folded-cube networks.

Fault-tolerance was incorporated into the above network through the technique of modular sparing. The cluster networks were made internally redundant by adding

spare nodes. The performance of this network was evaluated and compared to other hierarchical fault-tolerant networks. The fault-tolerance of this architecture was evaluated, and a reconfiguration strategy was proposed for this architecture. A mechanism for message routing in the 4cube-4spare basic block was also investigated.

8.2 Future Work

The 4cube-4spare architecture could be further refined so that it has the same mean internodal distance with a lesser number of links. This will decrease it's LP ratio and make it more competitive with the hypercube.

The network reliability model proposed in this work could be refined further to obtain a tighter bounds on the network reliability of the basic block. Also, this model could be validated by computing the exact reliability of the 4cube-4spare basic block.

A fault-tolerant routing algorithm has been developed for the 4cube-4spare basic block. This could be combined with the algorithm for the torus to get a routing algorithm for the architecture as a whole. Simulations can be done, under the assumptions of various traffic distributions to evaluate the dynamic performance of this architecture.

A complete performance analysis of the *HyperTorus* could be carried out. Some of the aspects which could be investigated are:

- Broadcasting, multicasting and generalized spanning trees.
- Embeddings of the hypercube and the mesh networks in the HyperTorus.
- Comparisons with other networks.

In this work, we have proposed a combination of the hypercube and the torus. This combination has been shown to have a constant degree and compares favorably with the hypercube when cost is used as a metric. It must be investigated to see if a similar performance improvement could be achieved by other graph combinations.

Bibliography

[1] J. Squire and S. M. Palais, "Programming and Design Considerations of a Highly Parallel Computer", in *Proc. AFIP Spring Joint Computing Conference*, 1963, pp. 395–400.

[2] C. L. Seitz, "The Cosmic Cube", *Communications of the ACM*, vol. 28, no. 1, pp. 7–19, January 1985.

[3] J. P. Hayes, "A Microprocessor Based Hypercube Supercomputer", *IEEE Micro*, vol. II, no. 4, pp. 7–19, October 1986.

[4] Y. Saad and M. A. Shultz, "Topological Properties of Hypercubes", Tech. Rep., Yale University, Dept. of Computer Science, 1985.

[5] J. R. Armstrong and F. G. Gray, "Fault Diagnosis in a Boolean n-cube Array of Microprocessors", *IEEE Transactions on Computers*, vol. 30, no. 8, pp. 587–590, August 1991.

[6] Christopher Rose, "Low Mean Internodal Distance Network Topologies and Simulated Annealing", *IEEE Transactions on Communications*, vol. 40, no. 8, pp. 1319–1326, August 1992.

[7] Christopher Rose, "Mean Internodal Distance in Regular and Random Multihop Networks", *IEEE Transactions on Communications*, vol. 40, no. 8, pp. 1310–1318, August 1992.

[8] Sivarama P. Dandamudi, *Hierarchical Hypercube Multicomputer Interconnection Networks*, Ellis Horwood series. Pitman, 1991.

[9] D. A. Reed and D. C. Grunwald, "The Performance of Multicomputer Interconnection Networks", *IEEE Computer*, vol. 20, no. 6, pp. 63–73, June 1987.

[10] Sieteng Soh and Suresh Rai, "Improved Lower Bounds on the Reliability of Hypercube Architectures", *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 4, pp. 364–378, April 1994.

[11] Nian-Feng Tzeng and Sizheng Wei, "Enhanced Hypercubes", *IEEE Transactions on Computers*, vol. 40, no. 3, pp. 284–294, March 1991.

[12] David A. Rennels, "On Implementing Fault Tolerance in Binary Hypercubes", in *Digest of Papers - FTCS16: 16th. Annual International Symposium on Fault Tolerant Computing*, July 1986, pp. 344–349.

[13] Siu-Cheung Chau and A. L. Liestman, "A Proposal for a Fault-Tolerant Binary Hypercube Architecture", in *19th. International Symposium on Fault-Tolerant Computing*, June 1989, pp. 323–330.

[14] Sultan and Melhem, "An Efficient Modular Spare Allocation Scheme and it's Application to Fault-Tolerant Binary Hypercubes", *IEEE Transactions on Parallel and Distributed Systems*, vol. 2, no. 1, pp. 117–126, January 1991.

[15] C. S. Yang, L. P. Zu, and Y. N. Wu, "A Reconfigurable, Modular Fault-tolerant Hypercube Architecture", *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 10, pp. 1018–1032, 1994.

[16] Prithviraj Banerjee and Michael Peercy, "Design and Evaluation of Hardware Strategies for Reconfiguring Hypercubes and Meshes under Faults", *IEEE Transactions on Computers*, vol. 43, no. 7, pp. 841–848, July 1994.

[17] Prithviraj Banerjee and Michael Peercy, "Design and Evaluation of Hardware Strategies for Reconfiguring Hypercubes and Meshes under Faults", Tech. Rep., Coordinated Science Lab, Illinois, 1992.

[18] A. L. N. Reddy et al, "I/O Embedding in Hypercubes", in *Proc. International Conference on Parallel Processing-Volume I.* August 1988, pp. 331–338, Pennsylvania State University.

[19] Abdul Hai Mohammed Abdullah, "Reliability of Modular Fault-Tolerant Hypercube Networks", Master's thesis, King Fahd University of Petroleum and Minerals, Dept. of Computer Engineering, 1995.

[20] Sivarama P. Dandamudi and Derek L. Eager, "Hierarchical Interconnection Networks for Multicomputer Systems", *IEEE Transactions on Computers*, vol. 39, no. 6, pp. 786–797, June 1990.

[21] Sivarama P. Dandamudi, "Performance Analysis of a Class of Hierarchical Hypercube Multicomputer Networks", *Performance Evaluation*, vol. 13, no. 3, pp. 159–179, June 1991.

[22] Qutaibah M. Malluhi and Magdy A. Bayoumi, "The Hierarchical Hypercube: A New Interconnection Topology for Massively Parallel Systems", *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 1, pp. 17–30, January 1994.

[23] J. Mohan Kumar and L. M. Patnaik, "Extended Hypercube: A Hierarchical Interconnection Network of Hypercubes", *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 1, pp. 45–57, January 1992.

[24] Chienhua Chen, Dharma P. Agrawal, and J. Richard Burke, "dBCube: A New Class of Hierarchical Multiprocessor Interconnection Networks with Area Efficient Layout", *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 12, pp. 1332–1344, December 1993.

[25] E. Ganesan and D. K. Pradhan, "The Hyper-deBruijn Multiprocessor Networks", in *Proc. 11th. Conference on Distributed Computing Systems*, May 1991, pp. 492–499.

[26] Peter Thomas Breznay and Mario Alberto Lopez, "A Class of Static and Dynamic Hierarchical Interconnection Networks", in *Proc. International Conference on Parallel Processing-Volume I*, June 1994, pp. 59–62.

[27] Yashovardhan R. Potlapalli and Dharma P. Agrawal, "HMIN: A New Method for Hierarchical Interconnection of Multiprocessors", in *Proc. International Conference on Parallel Processing*, 1993, pp. I303–I306.

[28] Kanad Ghose and Kiran Raghavendra Desai, "The HCN: A Versatile Interconnection Network based on Cubes", in *Proc. International Conference on Supercomputers*, November 1989, pp. 426–435.

[29] Mostafa H. Abd-El-Barr, M.A. Abdul Hai, and M. S. T. Benten, "Subcube Reliability of a Modular Fault-Tolerant Hypercube Architecture", in *Proc. ISCA International Conference on Parallel and Distributed Computing Systems*, September 1995, pp. 268–274.

[30] S. P. Dandamudi, "A Performance Comparison of Routing Algorithms for Hierarchical Hypercube Multicomputer Networks", in *Proc. International Conference on Parallel Processing*, August 1990, pp. 281–285.

[31] Peter Thomas Breznay and Mario Alberto Lopez, "A Class of Static and Dynamic Hierarchical Interconnection Networks", in *Proc. International Conference on Parallel Processing*, June 1993, pp. I–59 – I–62.

[32] Chienhua Chen, Dharma P. Agrawal, and J. Richard Burke, "dBCube: A New Class of Hierarchical Multiprocessor Interconnection Networks with Area Efficient Layout", *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 12, pp. 1332–1344, December 1993.

[33] M. L. Schlumberger, *De Bruijn Communication Networks*, PhD thesis, Stanford University, 1974.

[34] M. Imase and M. Itoh, "Design to Minimize Diameter on Building-Block Network", *IEEE Transactions on Computers*, vol. C-30, no. 6, pp. 439–442, June 1981.

[35] Sizheng Wei and Saul Levy, "Design and Analysis of Efficient Hierarchical Interconnection Networks.", in *Proc. Supercomputing'91*, November 1991, pp. 390–399.

[36] Paul Cull and Shawn M. Larson, "The Mobius Cubes", *IEEE Transactions on Computers*, vol. 44, no. 5, pp. 647–659, May 1995.

[37] A. El-Amawy and Shahram Latifi, "Properties and Performance of Folded Hypercubes", *IEEE Transactions on Parallel and Distributed Systems*, vol. 2, no. 1, pp. 31–42, July 1991.

[38] M. Abd-El-Barr, Feroze Daud, and K. M. Al-Tawil, "A Hierarchical Fault-Tolerant Interconnection Network", in *Proc. IEEE International Phoenix Conference on Computers and Communications*, March 27-29 1996, pp. 123–128.

[39] Khalid M. Al-Tawil and Dimiter R. Avresky, "Reconfiguration of Faulty Hypercubes", in *Proc. First European Dependable Comput. Conference*, October 1994, pp. 529–545.

[40] Khalid M. Al-Tawil, Dimiter R. Avresky, and D. Pradhan, "Fault Tolerance of Hypercubes using Spanning Trees", Research Report 93-055, Texas A&M University, Dept. of Computer Science., June 1993, Research Report.

[41] Abraham and Padmanabhan, "Reliability of the Hypercube", in *Proc. TENCON'93*, April 1993, pp. 142–145.

[42] Prasant Mohapatra and Chita R. Das, "On Dependability Evaluation of Mesh-Connected Computers", *IEEE Transactions on Computers*, vol. 44, no. 9, pp. 1073–1084, September 1995.

[43] T.K.Boehme, A. Kossow, and W. Preuss, "A generalization of consecutive-k-out-of-n:F systems", *IEEE Transactions on Reliability*, vol. 41, no. 9, pp. 451–457, September 1992.

[44] C. J. Colbourn, *The Combinatorics of Network Reliability*, Oxford University Press, New York, 1987.

[45] S. Rai and D. P. Agrawal, *Advances in Distributed System Reliability*, IEEE Computer Society Press, 1990.

[46] C. S. Yang, J. F. Wang, J. Y. Lee, and F. T. Boesch, "Graph Theoretic Reliability Analysis for the Boolean n-cube Networks", *IEEE Transactions on Circuits and Systems*, vol. 8, no. 8, pp. 1175–1179, September 1988.

[47] D. Bulka and J. B. Dugan, "A Lower Bound on the Reliability of an n-dimensional Hypercube", in *Proc. 9th. Symposium on Reliable Distributed Systems*, 1990, pp. 44–53.

[48] H. Sullivan and T. Brashkow, "A Large-scale, Homogenous, Fully-Distributed Parallel Machine", in *Proc. of the Fourth Symposium on Computer Architecture*, 1977, pp. 105–117.

[49] Youran Lan, "A Fault Tolerant Routing Algorithm in Hypercubes", in *Proc. International Conference on Parallel Processing-Volume III*, August 1994, pp. 163–166.

[50] W.J. Dally and C. L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks", *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547–553, May 1987.

Vita

- Feroze B. Daud
- Born in 1970, at Hyderabad, India.
- Received the Bachelor of Engineering degree in Computer Science and Engineering from Osmania University, Hyderabad in August, 1991.
- Worked as a Trainee Engineer in the R & D division at OMC Computers Ltd., Secunderabad, India from October, 1991 to October, 1993.
- Joined the Information and Computer Sciences Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, as a Research Assistant in November, 1993.
- Completed Masters Degree requirements in the Information and Computer Sciences Department in June 1996.